

Optical Wireless Communication Theory and Technology

Xizheng Ke

Coding Theory in Optical Wireless Communication Systems

Volume II



Science Press
Beijing



Springer

Optical Wireless Communication Theory and Technology

Series Editor

Xizheng Ke, School of Automation and Information Engineering, Xi'an University of Technology, Xi'an, Shaanxi, China

The book series Optical Wireless Communication Theory and Technology aims to introduce the key technologies and applications adopted in optical wireless communication to researchers of communication engineering, optical engineering and other related majors. The individual book volumes in the series are thematic. The goal of each volume is to give readers a comprehensive overview of how the theory and technology in a certain optical wireless communication area can be known. As a collection, the series provides valuable resources to a wide audience in academia, the communication engineering research community and anyone else who are looking to expand their knowledge of optical communication.

Xizheng Ke

Coding Theory in Optical Wireless Communication Systems

Volume II

Xizheng Ke
School of Automation and Information
Engineering
Xi'an University of Technology
Xi'an, Shaanxi, China

ISSN 2731-5967 ISSN 2731-5975 (electronic)
Optical Wireless Communication Theory and Technology
ISBN 978-981-97-2381-2 ISBN 978-981-97-2382-9 (eBook)
<https://doi.org/10.1007/978-981-97-2382-9>

Jointly published with Science Press

The print edition is not for sale in China mainland. Customers from China mainland please order the print book from: Science Press.

© Science Press 2024

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publishers, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publishers nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publishers remain neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.

The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

If disposing of this product, please recycle the paper.

Contents

1	Performance Analysis of Quasi-Pulse-Position Modulation	
	Methods	1
1.1	Pulse-Position Modulation in Optical Wireless Communication	1
1.1.1	Symbol-Structure Analysis of Various Modulation Modes	2
1.1.2	Basic Performance Analysis of Pulse-Position Modulation	8
1.2	Analysis of Time-Slot Error Rate	21
1.2.1	Analysis of Time-Slot Error Rate Under a Gaussian Channel Model	21
1.2.2	Analysis of the Slot Error Rate Under a Turbulent Channel Model	25
1.3	Channel Capacity	32
1.3.1	Transmission Capacity	33
1.3.2	Average Channel Capacity with Turbulent Channels	34
	References	46
2	Error Control Based on RS Codes	49
2.1	Basic Principles of Error-Control Coding	49
2.1.1	Basic Error-Control Methods	49
2.1.2	Error-Control Fundamentals [10, 11]	50
2.1.3	Error-Control Code Classification	51
2.2	Basic RS-Code Principles	52
2.2.1	Basic Concept of an RS Code [12, 13]	53
2.2.2	Properties of RS Codes	53
2.3	RS-Code Encoding Method	54
2.3.1	Galois-Field Arithmetic [1]	54
2.3.2	RS-Code Encoding Methods [14–26]	56
2.4	RS-Code Decoding Methods [27–29]	57

2.4.1	RS-Code Frequency-Domain Decoding Algorithms [27–29]	58
2.4.2	RS-Code Domain-Decoding Algorithm	61
2.5	Correlation Algorithms for RS-Code Decoding	61
2.5.1	Peterson–Gorenstein–Zierler Decoding Algorithm	62
2.5.2	Berlekamp–Massey (BM) Decoding Algorithm [23]	64
2.5.3	Qian’s Search Method	67
2.5.4	Forney’s Algorithm	69
2.5.5	Euclidean Decoding Algorithm [2]	69
2.6	Hardware Implementation of an RS(15, 9) Code-Encoding System	72
2.6.1	Numerical Operations in a Finite Field $GF(2^4)$	72
2.6.2	Implementation of the Coding Circuit [6, 7, 11]	75
2.6.3	Design of an RS(15, 9)-Code Decoding System	77
2.6.4	Shortened RS Code	83
2.7	Verilog HDL Implementation of Key Hardware Circuits	85
2.7.1	Verilog HDL Description of Key Circuits in the Encoder	85
2.7.2	Verilog HDL Description of Key Circuits in the Decoder	86
2.8	Analysis of Error-Correction Performance of RS Codes	87
2.8.1	Error Probability of RS Codes [35, 36]	88
2.8.2	RS Code Performance Under Burst Noise [35, 36]	90
	References	91
3	Error Control Based on Turbo Codes	93
3.1	Turbo-Code Encoding Theory	93
3.2	Convolutional Codes: Component Codes of Turbo Codes	95
3.2.1	Basic Concept and Coding Method of Convolutional Codes	95
3.2.2	Generation Matrix and Polynomial Description of Convolutional Codes	97
3.3	Turbo-Code Interleavers	102
3.3.1	Basic Interleaver Concepts	102
3.3.2	Effects of the Interleaver on Turbo-Code Performance	103
3.3.3	Typical Interleavers	104
3.3.4	Interleaver Properties	107
3.4	Decoding Turbo Codes	111
3.4.1	Turbo-Code Decoding Structure	111
3.4.2	Soft-Output Decoding Algorithm Based on a Posterior Probability	112
	References	123

4	Error Control Based on LDPC Codes	125
4.1	LDPC Code Overview	125
4.1.1	Description of LDPC Codes	125
4.1.2	LDPC-Code Loops	127
4.1.3	Classification of LDPC Codes	128
4.1.4	LDPC-Code Verification-Matrix Construction Method	132
4.2	Random Construction of LDPC Codes	133
4.2.1	Gallager's Random Construction Method	133
4.2.2	Mackay's Random Construction Method	135
4.2.3	RU Random Construction Method	136
4.3	Construction of Euclidean Geometric LDPC Codes	138
4.3.1	Basic Theory of Finite Fields	139
4.3.2	Euclidean Geometric LDPC Codes of the First Cyclic-Structure Type	141
4.3.3	Euclidean-Geometric LDPC Codes of the Second Cyclic-Structure Type	143
4.3.4	Constructing LDPC Codes from Clusters of Parallel Lines in Euclidean Geometry	145
4.4	LDPC-Code Decoding and Performance Estimation	146
4.4.1	LDPC Hard-Decision Bit-Flip Decoding	147
4.4.2	BP Iterative-Decoding Algorithm for LDPC Codes	149
4.4.3	Performance Estimation and Analysis of LDPC Codes	157
4.5	LDPC-Code Construction	164
4.5.1	π -Rotating LDPC-Code Coding Principle	164
4.5.2	π -Design of Rotation Matrices and Submatrices	166
4.5.3	Extending Non-Regularity π -Rotate LDPC Code	167
4.6	Performance Analysis of Euclidean Geometric LDPC Codes	171
4.6.1	Performance Comparison Between Uncorrected Codes and Corrected Codes	171
4.6.2	Performance Comparison of Euclidean-Geometric LDPC Codes with Different Bit Rates	172
4.6.3	Performance Comparison of LDPC Codes in Different Turbulent Channels	173
4.6.4	Performance Comparison with RS Codes	175
4.6.5	Performance Comparison of Different Numbers of Iterations	176
4.7	Hardware Implementation of an LDPC Code Encoder	177
4.7.1	General Block Diagram of the Encoder Implementation	177
4.7.2	Coding Implementation of π -Rotating LDPC Code	178
4.7.3	Pipeline Processing of the Coding	180
4.7.4	Coded ModelSim Simulation	181
4.8	LDPC-Code Decoding Implementation	182
4.8.1	General Block Diagram of the Decoder	182
4.8.2	Data-Input Module	182

4.8.3	Parallel Iterative-Decoding Module	183
4.8.4	Data-Output Module	185
	References	186
5	Research on Polar Codes in Optical Wireless Communication Systems	189
5.1	Polar Codes in Optical-Wireless Channels	189
5.1.1	Polar-Code Coding	189
5.1.2	Optical-Wireless Channel Model	196
5.1.3	Polarization of an Atmospheric-Turbulence Channel	199
5.1.4	Polar-Code Construction in a Weak Atmospheric-Turbulence Channel	204
5.2	Performance of Polarization Code in Wireless Optical Channel	205
5.2.1	SC Decoding Algorithm	205
5.2.2	Performance Analysis of Polar Codes Under PPM Technology	211
5.2.3	Performance of Polar Codes Under Subcarrier Modulation	215
5.2.4	Comparison Between Polar Codes and LDPC Codes	220
	References	220
6	Channel Measurements and Error-Control Experiments	223
6.1	Basic Laser-Transmission Theory in the Atmosphere	223
6.1.1	Energy Attenuation of a Laser Atmospheric Transmission [9–16]	224
6.1.2	Atmospheric-Turbulence Effect	228
6.1.3	Principle of an Experimental Measurement System [21–23]	234
6.2	Analysis of the Measurement Results of a Light-Intensity Experiment [21]	235
6.2.1	Daily Variation of the Light-Intensity Scintillation	236
6.2.2	Impact of Different Weather Conditions on an Optical-Communication Link	237
6.2.3	Spot-Jitter Analysis	240
6.3	Bit-Error Rate Simulation and Measurement-Result Analysis	242
6.3.1	Numerical Simulation of the Bit-Error Rate Under Different Weather Conditions	242
6.3.2	Bit-Error Rate Measurement	245
6.3.3	Error-Correction Performance Analysis	249
6.4	Analysis of the Influence of Meteorological Conditions on the System Bit-Error Rate	250
6.5	Attenuation of a Laser Signal in Rain	252
6.5.1	Single-Raindrop Forward Scattering [33–35]	253

6.5.2	Single-Sphere Particle Computer Simulation [36–42]	254
6.5.3	Attenuation of Sparse Rain Particles	257
	References	260
7	Adaptive Coding Based on Channel Estimation	263
7.1	Adaptive-Modulation Coding	264
7.1.1	Adaptive-Modulation Coding Strategy	264
7.1.2	Implementation of Adaptive-Modulation Coding	266
7.1.3	Analysis of the Adaptive-Modulation Coding Algorithm	267
7.2	Adaptive-Channel-Estimation Algorithm	270
7.2.1	Training-Sequence Design	271
7.2.2	Channel Estimation	273
7.2.3	Performance Simulation and Result Analysis	283
7.2.4	Improved Channel-Estimation Algorithm	284
7.3	Signal-to-Noise Ratio Estimation Algorithm	286
7.3.1	SNR Estimation Based on Subspace Autocorrelation	287
7.3.2	SNR Estimation Based on Noise-Variance Estimation	289
7.3.3	SNR Estimation Based on the Least-Squares Criterion	290
7.3.4	SNR Estimation Using a Correlation-Coefficient Solution	291
7.3.5	Performance Simulation and Discussion	293
7.3.6	Adaptive Judgment Based on Channel Estimation	293
7.3.7	Estimation-Error Analysis	297
7.4	Experimental Measurement and Simulation	298
	References	302
8	Time-Domain Equalization Based on Adaptive Filtering	305
8.1	Adaptive-Filtering Principle	305
8.1.1	Adaptive Filters	306
8.1.2	Application of Adaptive Filters	307
8.1.3	Adaptive-Filtering Algorithm	309
8.2	Analysis of LMS-Like Algorithms	313
8.2.1	LMS-Algorithm Analysis	313
8.2.2	Normalized LMS Algorithm	315
8.2.3	Simplified LMS Algorithm	317
8.2.4	Variable-Step-Size LMS Algorithm [10–28]	318
8.3	Adaptive Equalizers	331
8.3.1	Adaptive Linear Equalizers	331
8.3.2	Decision-Guided Adaptive Equalizer	333
8.3.3	Adaptive Fractional-Interval Equalizer	333
8.3.4	Blind Equalizers	335
8.4	Adaptive-Equalization Simulations of Wireless Laser Communications	338
8.4.1	Adaptive-Equalization Simulations	338

8.4.2	Analysis of Simulation Results	357
	References	359
9	Channel Blind Equalization Based on Higher-Order Statistics	363
9.1	Blind-Equalization Algorithm	363
9.1.1	Adaptive and Blind Equalization	363
9.1.2	Analysis of Blind-Equalization Algorithms for Higher-Order Statistics	367
9.1.3	Normalized Performance Simulation	375
9.1.4	Direct-Method Simulation Results	382
9.2	Analysis of the Bussgang Blind-Equalization Algorithm	389
9.2.1	Definition of the Bussgang Process	389
9.2.2	Mathematical Model of Bussgang's Algorithm	389
9.2.3	Analysis of the Bussgang Blind-Equalization Algorithm	391
9.3	Channel Equalization in Optical Wireless Communication	401
9.3.1	Simulation of the Combined DD Bussgang/CDLMS Algorithm for Blind Equalisation	402
9.3.2	Direct Method	404
9.3.3	Step Size and Parameter Selection	406
	References	409

Chapter 1

Performance Analysis of Quasi-Pulse-Position Modulation Methods



Many modulation methods exist for intensity modulation/direct detection (IM/DD), the earliest of which is on–off keying (OOK) [1]; however, it suffers from the influence of external disturbances. For this reason, pulse-position modulation (PPM) has been introduced. PPM has a strong anti-interference ability and high power efficiency and requires low average optical power to complete optical communication. Thus, it is suitable for optical communication [2]. We call the class of modulation methods derived from PPM and multiple-pulse position modulation (MPPM) quasi-pulse position modulation. This chapter compares the characteristics of quasi-pulse position modulation methods.

1.1 Pulse-Position Modulation in Optical Wireless Communication

The main modulation methods for IM/DD systems are OOK, PPM, digital pulse-interval modulation (DPIM), dual-headed PIM (DHPIM), etc. These modulation methods use lasers as information carriers to transmit information, and the receiving ends of these carriers detect optical pulses at equal intervals. The positions in time required to determine the transmitted information can be collectively referred to as pulse-like position modulation.

1.1.1 *Symbol-Structure Analysis of Various Modulation Modes*

(1) **On-off keying modulation**

On-off keying modulation is the most widely used and simplest modulation method for IM/DD systems. It uses the presence or absence of optical pulses to transmit the information. In a time slot, optical pulses represent a transmitted bit “1” and no optical pulses represent a transmitted bit “0”. OOK modulation has the advantages of simplicity, easy implementation, and low frequency-stability requirements for the laser light source; the disadvantages are low power utilization and poor anti-interference ability. When the background light is strong or the communication is at high speeds, this modulation method has difficulty ensuring the communication reliability, and it has limitations in giving full play to the bandwidth advantages of optical communication.

(2) **Pulse-position modulation**

Pulse-position modulation (PPM) is a one-to-one correspondence between a group of n -bit binary data and the position of the pulse signal in a time period composed of time slots [3]. It uses the position of the pulse in a time slot to represent the transmitted information. The position of the pulse in the time slot is determined using the transmitted binary data. The relationship between the n -bit binary data and the position of the pulse can be written as

$$L = 2^{n-1}m_1 + 2^{n-2}m_2 + \cdots + 2m_{n-1} + m_n. \quad (1.1)$$

Compared with OOK, PPM increases the number of time slots per frame, improves the power efficiency, enhances the anti-interference ability, and increases the demand for bandwidth. For a given number of modulation bits, the number of time slots for each frame of information modulated by PPM is the same, and there is only one optical pulse in a time slot in each frame. Therefore, to correctly demodulate the original information, it is necessary to determine the starting position of the frame, using frame and time-slot synchronization.

(3) **Multiple-pulse position modulation**

Multiple-pulse position modulation (MPPM) is a one-to-one mapping of the position of a binary bit stream of length M , and p optical pulses in a time period composed of N time slots. The modulation method can be recorded as (M, n, p) MPPM, where $n = \min(N)$ and $C_n^p \geq 2^M$.

Each frame of information in MPPM contains n time slots, p of which have optical pulses; that is, there are p optical pulses. Therefore, there are $C_n^p = n!/(p!(n-p)!)$ pulse combinations, which correspond to 2^M groups of data information. This combination represents the transmission ability of MPPM, which increases with an increase in the number of pulses p . Compared with PPM, MPPM has a much higher communication ability. For example, given the number of time slots $n = 16$, if PPM is used, only four bits of information can be transmitted, whereas (m, 2) MPPM and (m, 3) MPPM can transmit six bits and nine bits of information, respectively.

(4) Differential pulse-position modulation

The symbol structure of DPPM is similar to that of PPM. The transmitted information is still represented by the position of an optical pulse in the time slot; however, the “0” after the “1” in the symbol structure is removed, which saves unnecessary waiting time. Thus, the average symbol length is greatly shortened. Compared with PPM, DPPM shortens the average symbol length, improves the transmission capacity, has higher power and bandwidth efficiency, and does not require symbol synchronization during demodulation because the length of each symbol is different. However, because the symbol length in this modulation method is not fixed, the buffer can easily overflow during demodulation.

(5) Dual-duration pulse-position modulation

Dual-duration pulse-position modulation (DDPPM) is an improved form of PPM. The pulse width is varied, based on the PPM mode. The position and width of the pulse in the symbol are determined by the transmitted information.

The specific mapping method is as follows: when $k < 2^{M-1}$, the pulse is located at the $(k + 1)$ time slot, the width is $\alpha/2$ time slots, and the subsequent 2^{M-1} empty time slots are removed; when $k > 2^{M-1}$, the pulse is located at the $(k + 1 - 2^{M-1})$ time slot, the pulse width is α time slots, and the 2^{M-1} empty time slots in front of the PPM mapping are removed.

DDPPM has a fixed symbol length, like PPM, but it is nearly half that of PPM, which improves the transmission capacity and bandwidth utilization. Its other characteristics, such as bandwidth requirements, error performance, and channel capacity, are discussed and analyzed in the following chapters.

(6) Dual-amplitude pulse-position modulation

Dual-amplitude pulse-position modulation (DAPPM) is an improved form of pulse-position modulation. The symbol length is 2^{M-1} time slots, which is half that of PPM. Two amplitude signal pulses are used to distinguish the first and second halves of the information. For $k < 2^{M-1}$, the pulse amplitude is A , and the 2^{M-1} idle slots behind the PPM mapping are removed. Otherwise, the pulse amplitude is βA , and the 2^{M-1} idle slots in front of the PPM mapping are removed.

If the m -bit data group is written as $K = (m_0, m_1, \dots, m_{M-1})$, the mapping relationship of the pulse position is

$$S_z = \begin{cases} m_{M-1} + 2m_{M-2} + \cdots + 2^{M-2}m_1 + 2^{M-1}m_0, & k < 2^{M-1}; \\ m_{M-1} + 2m_{M-2} + \cdots + 2^{M-2}m_1 + 2^{M-1}m_0 - 2^{M-1}, & k \geq 2^{M-1}, \end{cases} \quad (1.2)$$

where S_z indicates that the pulse time slot is at the z th position. The symbol structures of DAPPM and PPM are similar; however, the pulse amplitude of the former changes and its symbol length is half that of the latter, which improves the bandwidth utilization and transmission capacity.

(7) Differential-amplitude pulse-position modulation

Differential-amplitude pulse-position modulation (DAPPM) is a new modulation method proposed by U. Sethakaset and T. A. Gulliver in 2004 [4]. This is a combination of pulse-amplitude modulation (PAM) and DPPM. The symbol length and pulse amplitude change with the information transmission. The variation range of the amplitude is $\{1, \dots, A\}$, and the variation range of the symbol length is $\{1, \dots, L\}$. The number of data bits contained in each frame is $M = \log_2(A \times L)$. Taking $M = 3$ bits as an example, the symbol mappings of OOK, PPM, DPPM, and DAPPM are shown in Table 1.1.

From the symbol-structure perspective, the symbol length and pulse amplitude of DAPPM mode are not fixed. When the number of modulation bits is fixed, DAPPM symbols can be expressed in many ways, and their demodulation is more difficult than all of the modulation modes mentioned previously.

(8) Shortened pulse-position modulation

Shortened pulse-position modulation (SPPM) is a new modulation method proposed by Mei Hong Sui, Xin Sheng Yu, and Zhang Guo Zhou in 2009, based on traditional PPM modulation [5]. It divides the transmitted binary data into two parts: the first one bit and the last $(M - 1)$ bits. The data of the first bit remains unchanged, and the data of the last $(M - 1)$ bits are modulated according to the PPM mapping method. The modulated data are obtained by combining the two parts of the data.

During demodulation, the two parts of the information are demodulated separately to obtain the original information, which is then combined. Assuming that the transmitted binary data are $M = (m_1, m_2, \dots, m_M)$, the information is modulated in

Table. 1.1 Symbolic mappings of OOK, PPM, DPPM, and DAPPM

OOK	PPM	DPPM	DAPPM ($L = 4, A = 2$)	DAPPM ($L = 2, A = 4$)
000	10,000,000	1	1	1
001	01,000,000	01	01	01
010	00,100,000	001	001	2
011	00,010,000	0001	0001	02
100	00,001,000	00,001	2	3
101	00,000,100	000,001	02	03
110	00,000,010	0,000,001	002	4
111	00,000,001	00,000,001	0002	04

two parts. After m_1 is modulated, $n_1, (n_2, \dots, n_M)$ remain. After PPM, (n_2, \dots, n_{2M-1}) remains. After combination, it becomes $n = (n_1, n_2, \dots, n_{2M-1})$. Taking $M = \text{three bits}$ as an example, the symbol-mapping relationship between PPM and SPPM is shown in Table 1.2.

(9) Separated double-pulse position modulation

Separated double-pulse position modulation (SDPPM) improves on the basis of MPPM with two pulses. The SDPPM modulation mode includes two optical pulses in each symbol. Assuming that the length of each symbol is N , there are C_N^2 pulse combinations. To avoid intersymbol interference (ISI), combinations of continuous pulses are removed when selecting the pulse combination. Therefore, the number of available pulse combinations is $C_{N-1}^2 - (N - 2)$ and meets $C_{N-1}^2 - (N - 2) \geq 2^M$.

(10) Overlapping pulse-position modulation

Overlapping pulse-position modulation (OPPM) is an improved form of pulse-position modulation. The symbol structure is similar to that of PPM, except that each time slot is equally divided into several small time slots. The width of the optical pulse remains unchanged; the starting position starts from a small time slot, and there is a small time slot between adjacent optical pulses, resulting in pulse overlap. Hence, it is called overlapping pulse-position modulation.

Let the length of each symbol be equally divided into p time slots of length τ , and each time slot be equally divided into q small time slots of length τ/q ; q is called the overlap coefficient. The starting position of the optical pulse is $t_k = (k - 1)\tau$, $k = 1, 2, \dots, J$. The relationship between J , p , and q can be expressed as

$$J = q(p - 1) + 1. \quad (1.3)$$

J is equivalent to the symbol length L in PPM modulation, and the number of information bits that can be transmitted by each OPPM symbol is $\log_2(pq - q + 1)$. When the number of time slots p is equal, the number of information bits transmitted by OPPM per symbol is higher than that of PPM. For example, if $p = 4$ and $q = 3$,

Table 1.2 Symbolic mappings of PPM and SPPM

Binary data	PPM (m_1, m_2, \dots, m_8)	SPPM (m_1, m_2, \dots, m_5)
000	10,000,000	01,000
001	01,000,000	00,100
010	00,100,000	00,010
011	00,010,000	00,001
100	00,001,000	11,000
101	00,000,100	10,100
110	00,000,010	10,010
111	00,000,001	10,001

then $J = 10$; PPM can transmit two bits of information and OPPM can transmit three bits of information. The greater q is, the more information each symbol of OPPM contains. However, it is more difficult to demodulate at the receiver.

(11) Digital pulse-interval modulation

Pulse-interval modulation (PIM) uses the space–time-slot interval between adjacent optical pulses to represent the transmitted information. The pulse position in the modulation symbol is fixed at the starting position of the symbol, which is called the starting pulse; it is followed by several protection time slots and K space–time slots representing the transmitted information. K is the decimal number corresponding to the transmitted binary data. It can be seen that the difference between PIM and PPM is that the information in PPM is represented by the position of the optical pulse in the information frame, whereas the information in PIM is represented by the number of space–time slots between two adjacent pulses.

Digital PIM (DPIM) is a type of PIM. The symbol structure is roughly the same as that of PIM; however, a protection time slot is added after the start pulse to avoid intersymbol crosstalk. When demodulating at the receiving end, this modulation method does not require symbol synchronization. It only needs to start counting when a pulse is detected and stop counting when the next pulse is detected. It then subtracts 1 from the calculated number of time slots to obtain the number of information time slots and thus, determines the original information. Compared with PPM, DPIM greatly simplifies the complexity of the system implementation.

(12) Double-headed pulse-interval modulation

DHPIM is an improved form of pulse-interval modulation. Compared with PIM, it changes the manner in which the width of the starting pulse and the information time slot correspond to the decimal value of the binary data. The DHPIM modulation mode is complex. Each symbol is composed of a head time slot and a subsequent space–time slot. The fixed length of the head time slot is $(\alpha + 1)$ time slots (α is a positive integer), which can be divided into two cases: an $\alpha/2$ time slot and an $(\alpha/2 + 1)$ time slot, corresponding to the A time slot and the one time slot.

k is the decimal number corresponding to the binary data and M is the number of bits of the modulation data. When $k < 2^{M-1}$, the head time slot is an $(\alpha/2)$ pulse time slot plus an $(\alpha/2 + 1)$ empty time slot, which is the protection time slot. The number of subsequent empty time slots is k , indicating the information to be transmitted. When $k \geq 2^{M-1}$, the head time slot is an α pulse time slot and a protection time slot, and the number of subsequent space–time slots is $(2^{M-1} - k)$.

DHPIM modulation adopts two optical pulses with different widths as the starting pulse, which clearly separates the first and second halves of the binary-information combination. The number of space–time slots representing information in the subsequent symbols is reduced, the number of redundant time slots is eliminated, the average symbol length is shortened, and the transmission capacity is improved. During demodulation, different start pulses can be used to mark the start of the

frame and distinguish whether the information is in the first or second half; thus, the DHPIM modulation mode has built-in symbol synchronization.

(13) Dual-pulse pulse-interval modulation

Dual-pulse pulse-interval modulation (DPPIM) is based on a modification of pulse-interval modulation, which changes the number and width of the pulses and the corresponding relation between the number of information time slots and binary data. The DPPIM symbol contains a fixed start pulse and a variable marking pulse. The width and amplitude of the starting pulse does not change. The marking pulse width and number of space-time slots between the two pulses vary with the change in the transmitted information. The average symbol length is $(2^{M-1} + \alpha)$ time slots. The width of the initial pulse is a time slot, indicating that the width of the pulse changes, owing to different information transmitted; this is set as the decimal number corresponding to the binary data.

When $k < 2^{M-1}$, the width of the marking pulse is $(\alpha/2)$ time slots, and the interval between the marking pulse and the starting pulse is k time slots. When $k \geq 2^{M-1}$, the width of the marking pulse is an α time slot, and the interval between the marking pulse and the starting pulse is $(k - 2^{M-1})$ time slots. Finally, several space-time slots are added after the marking pulse to ensure that the DPPIM symbol length is fixed. In addition, several space-time slots can be added between the starting pulse and marking pulse to reduce the influence of intersymbol crosstalk, and the average symbol length will also increase.

DPPIM adopts a fixed starting pulse, which provides this modulation method with built-in symbol synchronization. In addition, its symbol length is nearly half that of PPM, which improves the transmission capacity. Simultaneously, the symbol length is fixed, which overcomes the problems of buffer redundancy or overflow that may occur in modulation methods, such as DPIM and DHPIM, whose symbol length changes with the change in transmitted data.

(14) Double-amplitude pulse-interval modulation

DAPIM is obtained by changing the amplitude of the pulse, based on the pulse-interval modulation. The DAPIM symbol consists of a start pulse, a protection time slot, and m space-time slots. The amplitude of the start pulse changes as follows: when $k < 2^{M-1}$, the amplitude of the start pulse is A and the number of information time slots is $m = k$. When $k \geq 2^{M-1}$, the amplitude of the starting pulse is βA (β is a positive integer), and the number of information time slots is $m = k - 2^{M-1}$.

From the symbol-structure perspective, the symbol length of the DAPIM modulation mode is not fixed and varies with the transmitted information. Compared with PPM modulation, the symbol length is shortened by nearly half, which is equivalent to half the folding of the PPM frame. The two amplitudes of the starting pulse are used to distinguish the first and second halves of the information.

(15) Fixed-length digital pulse-interval modulation

Fixed-length digital pulse-interval modulation (FDPIIM), as the name suggests, has a fixed symbol length. It was proposed to address the disadvantage that the symbol

length of PIM was not fixed. The symbol length of FDPIM is fixed at $(2^M + 4)$ time slots. Each symbol includes a single time-slot start pulse, a protection time slot, k information time slots, a double time-slot marking pulse, and $(2^M - k)$ subsequent space-time slots. The first space-time slot after the marking pulse is the protection time slot, and the subsequent space-time slots do not represent any information; they simply ensure that the symbol length is fixed.

Although the symbol length of the FDPIM modulation mode is too long, that is, longer than PPM, its symbol length is fixed, and there are three optical pulses in each information frame, which is not necessarily worse than PPM in terms of transmission capacity. Previous pulse-interval modulation methods, such as DPIM, DHPIM, and DAPIM, shorten the symbol length and change the pulse width and amplitude to improve the performance, ignoring the problem of inconvenient demodulation caused by the unfixed symbol length. The FDPIM modulation method effectively solves this problem.

Although the symbol length of the DPPIM modulation method is fixed, it contains parameters that are not as clear as those of FDPIM. FDPIM also has a built-in symbol-synchronization capability. During demodulation, if it is determined that a single-slot pulse is received, the counter starts counting. If it is determined that a double-slot pulse is received, the counter stops counting. It then subtracts one from the remembered number to obtain the transmitted binary data. Finally, the original information is obtained through decoding.

(16) Fixed-length dual-amplitude pulse-interval modulation

Fixed-length dual-amplitude pulse-interval modulation (FDAPIM) is a combination of FDPIM and DAPIM and has a symbol structure similar to that of FDPIM mode. The difference lies in the method used to distinguish the starting pulse from the marking pulse. The initial pulse of FDPIM is the same as the marking pulse amplitude. The former is a single time slot, the latter is a double time slot, and the starting pulse of FDAPIM is the same as the marking pulse width. The former is A and the latter is βA . All other groups are the same.

Taking the number of modulation bits $M = 4$ as an example, the symbol-structure diagrams of each modulation mode are shown in Fig. 1.1.

In Fig. 1.1, α represents the pulse-width parameter and β represents the amplitude parameter. Among the above code words, fixed-length digital pulse-interval modulation (FDPIM) has the longest symbol length, followed by fixed-length dual-amplitude pulse-interval modulation (FDAPIM), followed by PPM, OOK, and DPPIM ($\alpha = 2$).

1.1.2 Basic Performance Analysis of Pulse-Position Modulation

In the following, some basic performance metrics of pulse-position modulation are discussed, such as the average symbol length, bandwidth requirement, and average transmission power.

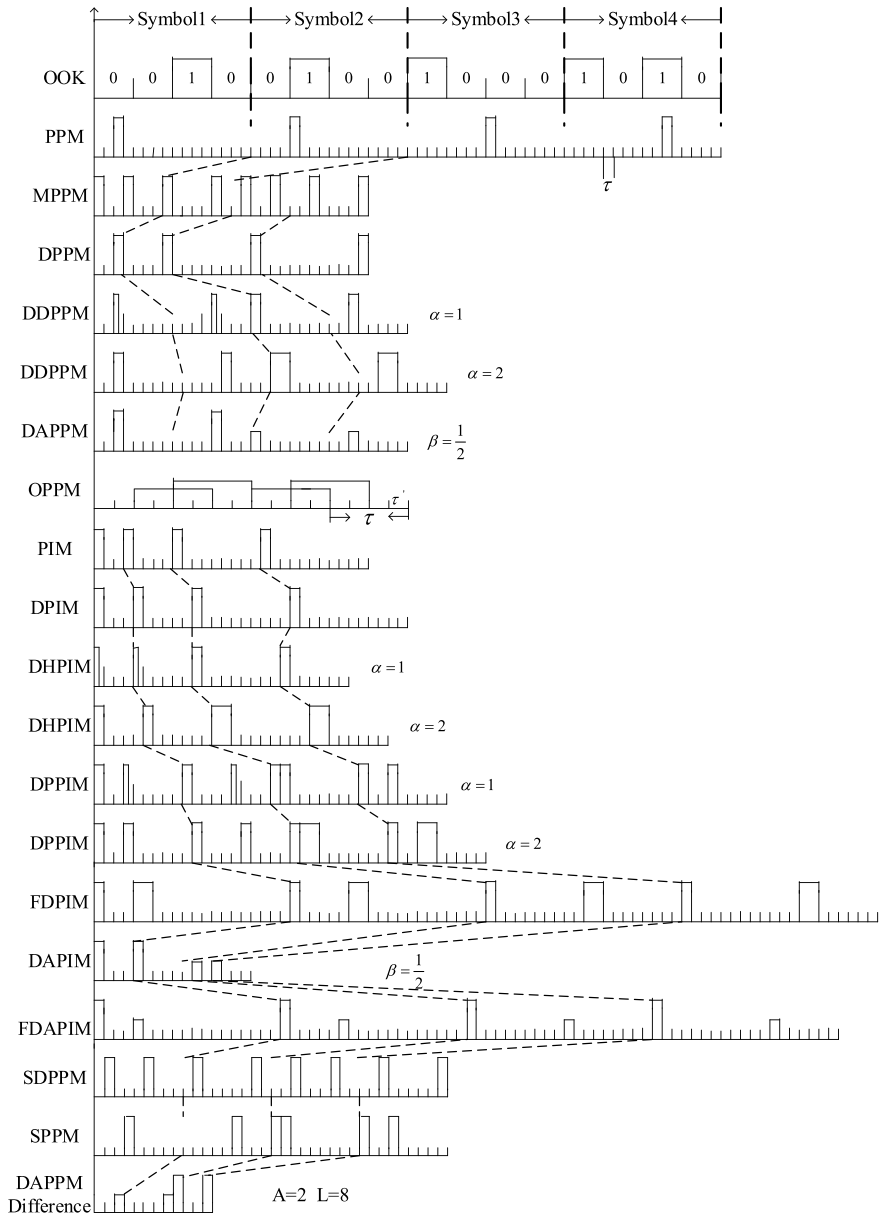


Fig. 1.1 Symbol structure of each modulation mode

(1) Average symbol length

The symbol length of each modulation mode refers to the number of time slots in an information frame. Some of the symbol lengths of the modulation modes described above are fixed and some vary with the transmitted data. We assume that the number of modulation bits per symbol is M and the width of each time slot is T_s .

- The symbol length of OOK is the number of modulation bits, according to the description of the symbol structure;
- The symbol length of PPM is $L_{PPM} = 2^M T_s$;
- The symbol length n of MPPM is related to the number of modulation bits m and the number of pulses p , which meet the relationship $C_n^p \geq 2^M$;
- SDPPM is based on multipulse position modulation with a pulse number of 2. Combinations of continuous pulses are removed, and the symbol length satisfies the relationship $C_{n-1}^2 - (n-2) \geq 2^M$;
- DPPM is obtained by removing the “0” time slot after the “1” in PPM modulation, which shortens the symbol length; however, the symbol length is not fixed. The minimum is T_s and the maximum is $2^M T_s$; thus, the average symbol length is $L_{DPPM} = (2^M + 1)T_s/2$.
- DAPPM is similar to PPM, except that the symbol length is shortened by two amplitudes, which is half of PPM, so $L_{DAPPM} = 2^{M-1} T_s$;
- DDPPM uses optical pulses of two widths to distinguish the first and second halves of information. Compared with PPM, its symbol length is also shortened by nearly half and fixed as $L_{DDPPM} = (2^{M-1} + \alpha - 1)T_s$;
- SPPM divides the transmitted M -bit binary data into two parts: the first bit and the last $(M - 1)$ bits. During modulation, the former part remains unchanged and the latter part is modulated according to PPM mode. The symbol length is also fixed, which is $L_{SPPM} = (2^{M-1} + 1)T_s$.
- DPIM uses the number of time slots between adjacent pulses to represent the transmitted information. This number of time slots is the decimal value corresponding to the transmitted binary data, so the symbol length is not fixed; the minimum is $2T_s$ and the maximum is $(2^M + 1) T_s$, so the average symbol length is $L_{DPIM} = (2^M + 3) T_s/2$;
- DHPIM also uses the number of time slots between adjacent pulses to represent information. It uses optical pulses of two widths to distinguish between the first and second halves of the information. Compared with DPIM modulation, the symbol length is shortened by nearly half. The symbol length is not fixed, the minimum is $(\alpha + 1) T_s$, the maximum is $(2^{M-1} + \alpha) T_s$, and the average symbol length is $L_{DHPIM} = (2^{M-1} + 2\alpha + 1) T_s/2$.
- DPPIM has two optical pulses in each symbol: a fixed starting pulse and a variable marking pulse. The information is represented by the number of time slots between the two pulses, and the average symbol length is $L_{DPPIM} = (2^{M-1} + \alpha) T_s$.
- DAPIM is an improvement of DPIM. Two amplitudes are adopted to shorten the average symbol length. $L_{DAPIM} = (2^{M-1} + 3) T_s/2$;
- The symbol length of FDPIM is the longest of these modulation methods, fixed as $L_{FDPIM} = (2^M + 4) T_s$;

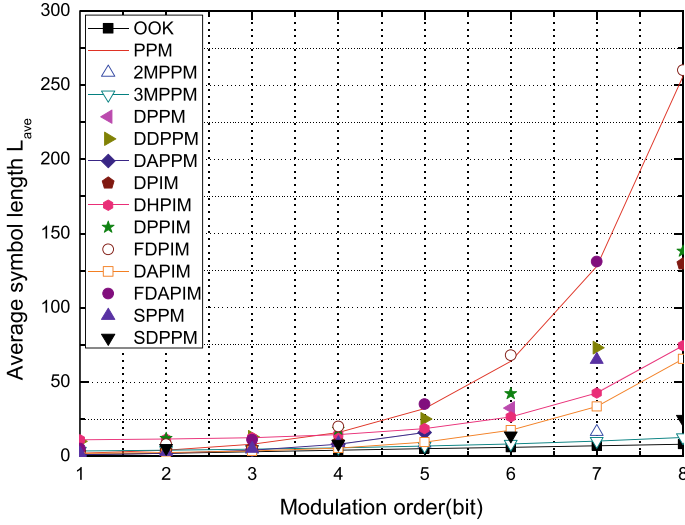


Fig. 1.2 Average symbol length of each modulation mode

- The symbol length of FDAPIM is second only to FDPIM, and the average symbol length is $L_{FDAPIM} = (2^M + 3) T_s$.

The average symbol lengths of the different modulation modes were compared by numerical simulation with MATLAB; the results are shown in Fig. 1.2.

Figure 1.2 shows the average symbol lengths for the various modulation modes. To more clearly distinguish the average symbol-length curves of different modulation modes in the figure, the pulse-width parameter was taken as 10.

It can be seen from the figure that when the modulation order M is very small, there is little difference in the average symbol lengths of these modulation methods. With the increase in M , the average symbol lengths of FDPIM, FDAPIM, and PPM increase the fastest; they are longer than those of the other modulation methods. The symbol lengths of these three are very close, followed by DPPIM and DDPPM, which are similar.

The average symbol lengths of SPPM, DPIM, DPPM, and DAPPM are also very close, and are almost half as short as the first three. The average symbol lengths of DHPIM and DAPIM are approximately twice as long as those of the first few. SDPPM, 2MPPM (MPPM with two pulses) and 3MPPM (MPPM with three pulses), and OOK are the smallest.

When the pulse-width parameter is 1, the average symbol lengths of DPPIM and SPPM are the same; DDPPM and DAPPM are the same; and DHPIM and DAPIM are the same. Therefore, for modulation modes with parameters, the average symbol length increases with an increase in the parameters.

(2) Bandwidth requirements and utilization

Optical wireless communication systems require a certain amount of bandwidth to transmit information, and the smaller the bandwidth, the better. We assume that the transmission bit rate of the source is R_b bits/s and each symbol sends M bits of information.

If calculated using the first zero point of the power spectrum, $B_{OOK} \approx R_b$. The bandwidth required for OOK can be considered as the reciprocal of the pulse width; that is, $B_{OOK} = 1/T_b = R_b$. T_b is the OOK modulation slot width, and $T_b = 1/R_b$. Therefore, the bandwidth required by the modulation mode can be defined as $B = 1/T_s$, and the corresponding bandwidth utilization is defined as $\eta = R_b/B$ [6].

For different modulation modes, the time required to send a symbol should be the same when the source bit rate is the same. Therefore, for PPM, $2^M T_{PPM} = MT_b$, where T_{PPM} is the time-slot width of the PPM modulation mode. The bandwidth demand and utilization of PPM are respectively as follows:

$$B_{PPM} = \frac{1}{T_{PPM}} = \frac{2^M}{M} B_{OOK} \quad (1.4)$$

$$\eta_{PPM} = \frac{M}{2^M}. \quad (1.5)$$

The relationship between the symbol length n , number of pulses in the symbol p , and number of transmitted information bits M of the MPPM modulation mode satisfies the formula $C_n^p > 2^M$. According to the definition formula, the bandwidth demand and utilization of MPPM are, respectively,

$$B_{MPPM} = \frac{1}{T_{MPPM}} = \frac{n}{M} B_{OOK} \quad (1.6)$$

$$\eta_{MPPM} = \frac{M}{n}. \quad (1.7)$$

The average symbol length of DPPM is $L_{DPPM} = (2^M + 1)/2$. From $((2^M + 1)/2) T_{DPPM} = MT_b$, the bandwidth demand and utilization required for DPPM are, respectively,

$$B_{DPPM} = \frac{1}{T_{DPPM}} = \frac{2^M + 1}{2M} B_{OOK} \quad (1.8)$$

$$\eta_{DPPM} = \frac{2M}{2^M + 1}. \quad (1.9)$$

The average symbol length of DAPPM is $L_{DAPPM} = 2^{M-1}$. With the same bit rate and same number of transmitted bits, $2^{M-1} T_{DAPPM} = MT_b$, so the bandwidth demand and utilization required for DAPPM are, respectively,

$$B_{DAPPM} = \frac{1}{T_{DAPPM}} = \frac{2^{M-1}}{M} B_{OOK} \quad (1.10)$$

$$\eta_{DAPPM} = \frac{M}{2^{M-1}}. \quad (1.11)$$

The symbol length of DDPPM is $L_{DDPPM} = 2^{M-1} + \alpha - 1$, and the time-slot width of DDPPM obtained from $(2^{M-1} + \alpha - 1) T_{DDPPM} = MT_b$ is $T_{DDPPM} = MT_b / (2^{M-1} + \alpha - 1)$ because DDPPM uses two different pulse widths: the $\alpha/2$ and α time slots. To meet the bandwidth requirements, the minimum pulse duration should be selected; therefore, the pulse-width requirements and bandwidth utilization of the DDPPM method are, respectively,

$$B_{DDPPM} = \frac{1}{\frac{\alpha}{2} T_{DDPPM}} = \frac{2^M + 2\alpha - 2}{\alpha M} B_{OOK} \quad (1.12)$$

$$\eta_{DDPPM} = \frac{\alpha M}{2^M + 2\alpha - 2}. \quad (1.13)$$

The symbol length of SPPM is $L_{SPPM} = 2^{M-1} + 1$, and from $(2^{M-1} + 1) T_{SPPM} = MT_b$, the pulse-width requirement and bandwidth utilization rate of the SPPM mode are, respectively,

$$B_{SPPM} = \frac{1}{T_{SPPM}} = \frac{2^{M-1} + 1}{M} B_{OOK} \quad (1.14)$$

$$\eta_{SPPM} = \frac{M}{2^{M-1} + 1}. \quad (1.15)$$

The symbol length of the DPIM modulation method is not fixed, and the average symbol length is $L_{DPIM} = (2^M + 3)/2$, which is represented by $((2^M + 3)/2) T_{DPIM} = MT_b$. The required bandwidth and bandwidth utilization rate of DPIM are, respectively,

$$B_{DPIM} = \frac{1}{T_{DPIM}} = \frac{2^M + 3}{2M} B_{OOK} \quad (1.16)$$

$$\eta_{DPIM} = \frac{2M}{2^M + 3}. \quad (1.17)$$

The symbol length of DHPIM is not fixed, the minimum is $(\alpha + 1)$ time slots, the maximum is $(2^{M-1} + \alpha)$ time slots, and the average symbol length is $L_{DHPIM} = (2^{M-1} + 2\alpha + 1)/2$. The time used to transmit the same amount of data using the same transmission rate is the same, that is, $(2^{M-1} + 2\alpha + 1)/2 = MT_b$. The time-slot width of the DHPIM method is $T_{DHPIM} = 2MT_b / (2^{M-1} + 2\alpha + 1)$ because DHPIM uses two optical pulses of different widths: an $\alpha/2$ time slot and α time slot. To meet

the demand, the minimum pulse duration should be selected; thus, the bandwidth demand and utilization of DHPIM are, respectively,

$$B_{DHPIM} = \frac{1}{\frac{\alpha}{2}T_{DHPIM}} = \frac{2^{M-1} + 2\alpha + 1}{\alpha M} B_{OOK} \quad (1.18)$$

$$\eta_{DHPIM} = \frac{\alpha M}{2^{M-1} + 2\alpha + 1}. \quad (1.19)$$

Like DHPIM, DPPIM also has two types of optical pulse widths, and its required bandwidth is determined by an optical pulse with an $\alpha/2$ time-slot pulse width. The symbol length of DPPIM is $L_{DPPIM} = 2^{M-1} + \alpha$, and $T_{DPPIM} = MT_b(2^{M-1} + \alpha)$ is obtained from $(2^{M-1} + \alpha)T_{DPPIM} = MT_b$, and the required bandwidth and bandwidth utilization rate of DPPIM are, respectively,

$$B_{DPPIM} = \frac{1}{\frac{\alpha}{2}T_{DPPIM}} = \frac{2^M + 2\alpha}{\alpha M} B_{OOK} \quad (1.20)$$

$$\eta_{DPPIM} = \frac{\alpha M}{2^M + 2\alpha}. \quad (1.21)$$

The average symbol length of DAPIM is $L_{DAPIM} = (2^{M-1} + 3) / 2$. From $(2^{M-1} + 3)T_{DAPIM} / 2 = MT_b$, the bandwidth requirement and bandwidth utilization of DAPIM modulation are, respectively,

$$B_{DAPIM} = \frac{1}{T_{DAPIM}} = \frac{2^{M-1} + 3}{2M} B_{OOK} \quad (1.22)$$

$$\eta_{DAPIM} = \frac{2M}{2^{M-1} + 3}. \quad (1.23)$$

FDPIIM and FDAPIM have the longest symbol lengths among these modulation methods, and the symbol length is fixed in pulse-interval modulation. The width of the pulse in the symbol is the same, that is, a time slot. The required bandwidth and bandwidth utilization rates are, respectively,

$$B_{FDPIIM} = \frac{1}{T_{FDPIIM}} = \frac{2^M + 4}{M} B_{OOK} \quad (1.24)$$

$$\eta_{FDPIIM} = \frac{M}{2^M + 4} \quad (1.25)$$

$$B_{FDAPIM} = \frac{1}{T_{FDAPIM}} = \frac{2^M + 3}{M} B_{OOK} \quad (1.26)$$

$$\eta_{FDAPIM} = \frac{M}{2^M + 3}. \quad (1.27)$$

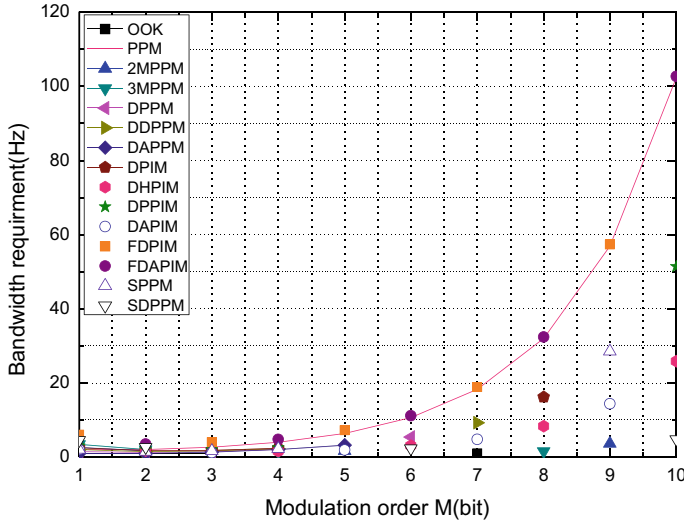


Fig. 1.3 Bandwidth requirements for different modulation methods

The expression of the bandwidth requirement and bandwidth utilization of SDPPM is the same as that of MPPM; however, the expression of symbol length n is different, and n satisfies $C_{n-1}^2 - (n-2) \geq 2^M$.

Figures 1.3 and 1.4 show the comparisons of the bandwidth requirements and bandwidth utilization of various modulation methods by the numerical simulation of Eqs. (1.4)–(1.27).

Figures 1.3 and 1.4 show a comparison of the bandwidth requirements and bandwidth utilization of various modulation modes. The pulse-width parameter α was set to 2 and the information bit rate was 1.

As shown in Fig. 1.3, the bandwidth requirements of each modulation method increase with the number of modulation bits. The bandwidth requirements of FDPIM, FDAPI, and PPM are the largest, followed by DPPIM, SPPM, DPIM, DAPPM, DPPM, and DDPPM. The bandwidth requirements of these six modulation methods are very close, and tend to be the same as the number of modulation bits increases. For DHPIM and DAPIM, the bandwidth requirements tend to be the same, as the number of modulation bits increases; finally, SDPPM, 2MPPM, 3MPPM, and OOK require the least bandwidth.

When the pulse-width parameter α is 1, the bandwidth requirement of DDPPM is the same as that of PPM, that of DHPIM is very close to that of SPPM, and that of DPPIM is very close to that of FDAPI. It can be seen that the bandwidth requirements of the modulation schemes with varying pulse widths decrease as parameter α increases. As shown in Fig. 1.4, the bandwidth utilization of each modulation method decreases with an increase in the number of modulation bits.

DAPPM and OPPM differ from those mentioned earlier. The symbol length of DAPPM is not fixed, the minimum is 1, the maximum is L , the average symbol length

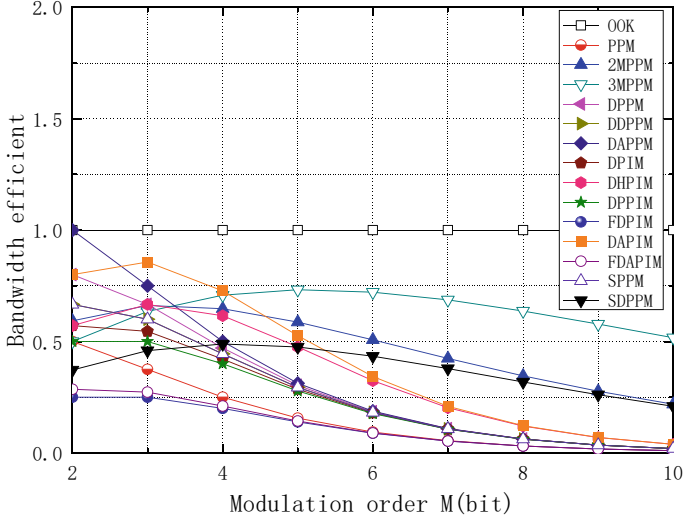


Fig. 1.4 Bandwidth utilization of different modulation methods

is $(L + 1)/2$, and the bandwidth requirement is

$$B_{DAPPM} = \frac{L + 1}{2M} B_{OOK} = \frac{L + 1}{2 \log_2(A \times L)} B_{OOK}. \quad (1.28)$$

It can be seen from Eq. (1.15) that the bandwidth requirement of DAPPM is related to its symbol length and pulse amplitude, and both the symbol length and pulse amplitude vary, satisfying the relationship. The relationship between the bandwidth requirement of DAPPM and the symbol length is shown in Fig. 1.5. As can be seen from the figure, the bandwidth requirement of DAPPM increases with increasing symbol length, and decreases with increasing pulse amplitude.

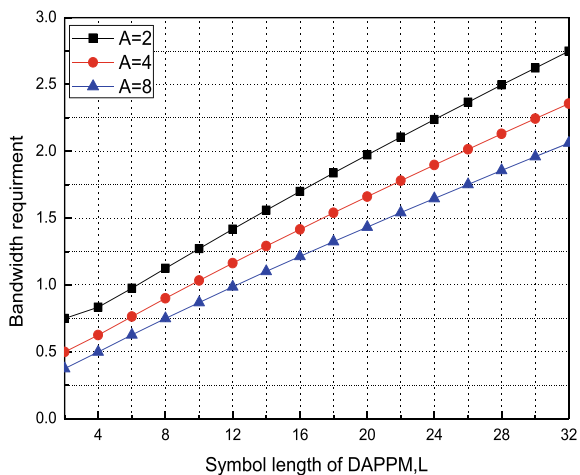
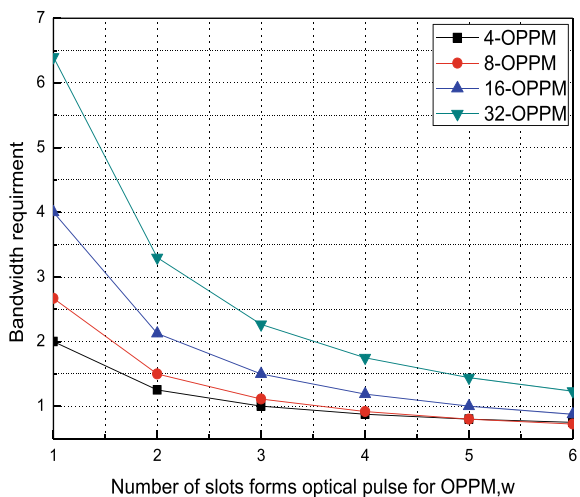
The OPPM method divides each time slot equally into several small slots in the symbol structure of PPM, and the width of each pulse remains unchanged. Assuming that the total number of time slots is n and the overlap coefficient is w [7],

$$\frac{n}{w} \cdot T_{OPPM} = \log_2(n - w + 1) T_b. \quad (1.29)$$

The bandwidth requirement of OPPM is

$$B_{OPPM} = \frac{1}{T_{OPPM}} = \frac{n}{w \log_2(n - w + 1)} B_{OOK}. \quad (1.30)$$

The relationship between the OPPM bandwidth requirement and overlap coefficient is shown in Fig. 1.6. The OPPM bandwidth requirement decreases with an increasing overlap coefficient and increases with J .

Fig. 1.5 DAPPM bandwidth requirements**Fig. 1.6** Bandwidth requirements for OPPM modulation

(3) Average transmit power

In an optical wireless communication system, a pulse-like position-modulation method can be considered as sending “0” and “1” sequences with equal probability. When sending a “0” sequence, no power is required, and when sending a “1” sequence, peak power P_c is required. Therefore, for a given peak transmit power P_c , the average transmit power can simply be determined as the probability of sending a “1” multiplied by this power; that is, $P_{ave} = p_1 P_c$. Then, $P_{ave-OOK} = P_c/2$ and $P_{ave-PPM} = P_c/2^M$.

The symbol length n , number of modulation bits M , and number of pulses p in the MPPM mode satisfy the relationship $C^p_n \geq 2^M$. According to the calculation formula, the average transmit power of MPPM is

$$P_{ave-MPPM} = \frac{p}{n} P_c. \quad (1.31)$$

The average symbol length of DPPM is $L_{DPPM} = (2^M + 1) / 2$; there is only one optical pulse in each symbol, and the average transmit power is

$$P_{ave-DPPM} = \frac{2}{2^M + 1} P_c. \quad (1.32)$$

The symbol length of DDPPM is fixed as $L_{DDPPM} = 2^{M-1} + \alpha - 1$, and two optical pulses with different pulse widths may appear in each symbol: an $\alpha/2$ time slot and α time slot. Then, the average pulse in each symbol is $3\alpha/4$; hence, the average transmit power is

$$P_{ave-DDPPM} = \frac{3\alpha}{2^{M+1} + 4\alpha - 4} P_c. \quad (1.33)$$

The symbol length of DAPPM is $L_{DAPPM} = 2^{M-1}$, and the average number of optical pulses in each symbol is $(1 + \beta)/2$; therefore, the average transmit power is

$$P_{ave-DAPPM} = \frac{1 + \beta}{2^M} P_c. \quad (1.34)$$

The average symbol length of SPPM is $L_{SPPM} = 1 + 2^{M-1}$ and the average number of pulses in each symbol is $3/2$. Therefore, the average transmit power is

$$P_{SPPM} = \frac{3}{2^M + 2} P_c. \quad (1.35)$$

The number of optical pulses in the SDPPM symbol is 2, the symbol length is n , the relationship $C_{n-1}^2 - (n - 2) \geq 2^M$ is satisfied, the minimum value satisfying this relationship is taken, and the average transmit power is

$$P_{ave-SDPPM} = \frac{2}{n} P_c. \quad (1.36)$$

The average symbol length of DPIM is $L_{DPIM} = (2^M + 3)/2$. Only one optical pulse appears in each symbol, so the average transmit power of DPIM modulation can be obtained as

$$P_{ave-DPIM} = \frac{2}{2^M + 3} P_c. \quad (1.37)$$

The average number of optical pulses in each symbol of DHPIM is $(\alpha/2 + \alpha)/2 = 3\alpha/4$, the average symbol length is $L_{DHPIM} = (2^{M-1} + 2\alpha + 1)/2$, and the average transmit power is

$$P_{DHPIM} = \frac{3\alpha}{2^M + 4\alpha + 2} P_c. \quad (1.38)$$

The marked pulse in the DPPIM symbol has two forms: its pulse width is an $\alpha/2$ time slot or an α time slot. The average number of pulses is $(1 + 3\alpha/4)$, and the average transmit power is

$$P_{ave-DPPIM} = \frac{4 + 3\alpha}{2^{M+1} + 4\alpha} P_c. \quad (1.39)$$

The amplitude of the optical pulse in the DAPIM modulation symbol has two forms: A and βA , the average number of pulses is $(1 + \beta)/2$, and the average symbol length is $L_{DAPIM} = (2^{M-1} + 3)/2$, so the average transmit power of the modulation method is

$$P_{ave-DAPIM} = \frac{1 + \beta}{2^{M-1} + 3} P_c. \quad (1.40)$$

The symbol length of the FDPIM mode is fixed as $L_{FDPIM} = 2^M + 4$, and there are three optical pulses in each symbol; then, the average transmit power is

$$P_{ave-FDPIM} = \frac{3}{2^M + 4} P_c. \quad (1.41)$$

The symbol length of FDAPIM is one time slot less than that of FDPIM. The amplitudes of the initial pulse and the marked pulse are A and βA , respectively, the average number of pulses is $(1 + \beta)/2$, and the average transmit power is

$$P_{ave-FDAPIM} = \frac{1 + \beta}{2^M + 3} P_c. \quad (1.42)$$

Equations (1.31)–(1.42) were numerically simulated and the average transmit power of the different modulation methods are shown in Fig. 1.7.

Figure 1.7 shows the relationship between the average transmit power and modulation order when the peak transmit power is constant. The pulse-width parameter α was set to 1 and the amplitude parameter β was set to 2. It can be observed from the figure that the average transmit power decreases with an increase in the modulation order. When $M \leq 3$, the average transmit power between the modulation methods follows no certain rules; when $M > 3$, the largest is 3MPPM, followed by 2MPPM, SDPPM, and DAPIM; the smallest is PPM.

A systematic summary of the average symbol length, bandwidth requirements, and average transmit power of the various modulation methods is shown in Table 1.3.

Fig. 1.7 Average transmit power of different modulation methods

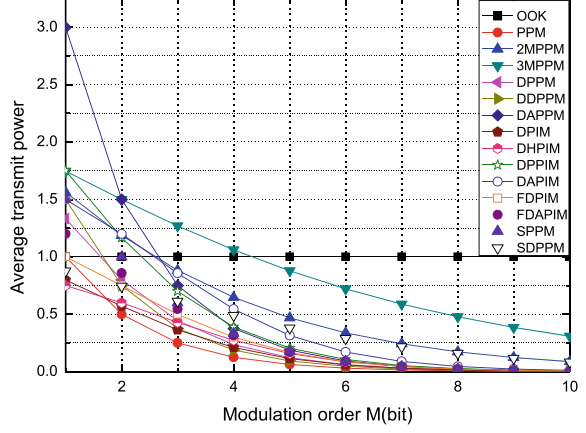


Table. 1.3 Average symbol length, bandwidth requirement, and average transmit power of each modulation method

Modulation	Average symbol length	Bandwidth requirements	Average transmit power
OOK	M	R_b	$P_c/2$
PPM	2^M	$2^M R_b/M$	$P_c/2^M$
MPPM	n (contains p pulses)	nR_b/M	pP_c/n
DPPM	$\frac{2^M+1}{2}$	$\frac{2^M+1}{2M} R_b$	$\frac{2}{2^M+1} P_c$
DDPPM	$2^{M-1} + \alpha - 1$	$\frac{2^M+2\alpha-2}{\alpha M} R_b$	$\frac{3\alpha}{(2^M+1+4\alpha-4)}$
DAPPM	2^{M-1}	$\frac{2^{M-1}}{M} R_b$	$\frac{1+\beta}{2^M} P_c$
PIM	$\frac{2^M+1}{2}$	$\frac{2^M+1}{2M} R_b$	$\frac{2}{2^M+1} P_c$
DPIM	$\frac{2^M+3}{2}$	$\frac{2^M+3}{2M} R_b$	$\frac{2}{2^M+3} P_c$
DHPIM	$\frac{2^{M-1}+2\alpha+1}{2}$	$\frac{2^{M-1}+2\alpha+1}{\alpha M} R_b$	$\frac{3\alpha}{(2^M+4\alpha+2)}$
DPPIM	$2^{M-1} + \alpha$	$\frac{2^M+2\alpha}{\alpha M} R_b$	$\frac{4+3\alpha}{2^M+1+4\alpha} P_c$
DAPIM	$\frac{2^{M-1}+3}{2}$	$\frac{2^{M-1}+3}{2M} R_b$	$\frac{1+\beta}{2^{M-1}+3} P_c$
FDPIM	$2^M + 4$	$\frac{2^M+4}{M} R_b$	$\frac{3}{2^M+4} P_t$
FDAPIM	$2^M + 3$	$\frac{2^M+3}{M} R_b$	$\frac{1+\beta}{2^M+3} P_c$
SPPM	$1 + 2^{M-1}$	$\frac{2^{M-1}+1}{M} R_b$	$\frac{3}{2^M+2} P_c$
SDPPM	n_s	$n_s R_b/M$	$2P_c/n_s$

1.2 Analysis of Time-Slot Error Rate

In an optical wireless communication system, the information sent by the source is coded and modulated and converted into a set of information sequences composed of multiple time slots. Each time slot takes a value of “1” or “0”, according to the presence or absence of optical pulses, and is then loaded into the laser and emitted through the optical antenna.

Because a laser transmission in the atmosphere is affected by factors such as turbulence, background-light noise, and path loss, the information will be distorted or lost, and the receiving end will receive incorrect information. The slot error rate refers to the probability of errors occurring in the time slots in each symbol information; that is, the sum of the probability of sending “1” and receiving “0” and the probability of sending “0” and receiving “1”. This section analyzes the slot error rate performance of various modulation methods in Gaussian and turbulent channels.

1.2.1 Analysis of Time-Slot Error Rate Under a Gaussian Channel Model

The channel model of an optical wireless communication system is related to the background light. When the background light is very weak, the received signal follows a Poisson distribution, whereas in the case of strong background-light interference, the received signal follows a Gaussian distribution. In the ideal case, the mathematical model of a Gaussian channel is [8]

$$y(t) = \int_{-\infty}^{+\infty} x(\tau)h(t - \tau) + n(t). \quad (1.43)$$

$y(t)$ is the output current at the receiving end, $x(t)$ is the optical power of the transmitted signal, $h(t)$ is the channel-multipath fading factor, and $n(t)$ is white Gaussian noise. In a traditional channel, $x(t)$ represents the amplitude of the signal, which satisfies the following expression:

$$x(t) \geq 0, \quad \lim_{T \rightarrow \infty} \frac{1}{2T} \int_{-T}^T x(t)dt \leq P, \quad (1.44)$$

where P is the average transmit power of the signal.

The digital modulation-system model of an optical wireless communication system is shown in Fig. 1.9. To facilitate the analysis, the following assumptions are made [9]:

- (1) The transceivers are accurately aligned without considering jitter;
- (2) there is no multipath effect;
- (3) only the influence of background-light noise is considered;

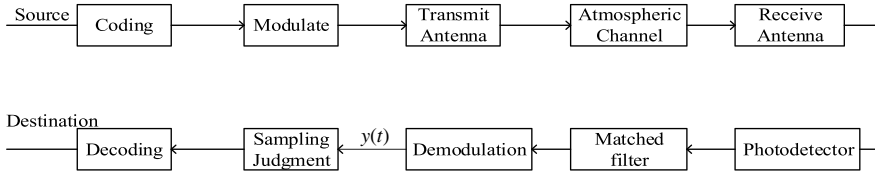


Fig. 1.8 Digital-modulation optical-communication system model

- (4) artificial light is not considered, and
- (5) the transmitter and receiver front ends are not limited by bandwidth.

The signal sent by the source is coded and modulated, loaded onto the laser beam, and sent out through the optical transmitting antenna. The signal received at the receiving end is converted into an electrical signal by a photodetector, and a matched filter is then used to amplify the signal to suppress noise. The modulated signal is sent to the decoder after sampling and judgment, and the original signal is obtained after decoding.

As shown in Fig. 1.8, the dominant noise $n(t)$ in the channel is white Gaussian noise with a mean of zero and variance σ_n^2 . Let $y(t)$ in the figure be $\sqrt{P_r} + n(t)$ when a pulse “1” is sent, and $n(t)$ when a pulse “0” is sent because the noise $n(t)$ is a Gaussian random variable: [10]

$$y(t) = \begin{cases} \sqrt{P_r} + n(t), & \text{transmit '0'} \\ n(t), & \text{transmit '1'} \end{cases} \quad (1.45)$$

is also a Gaussian variable, where P_r is the peak power of the signal at the arbiter input.

$$f_1(y) = \frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{(y-\sqrt{P_r})^2}{2\sigma_n^2}} \quad (1.46)$$

The probability-density function of the received signal y when the signal “0” is sent is

$$f_0(y) = \frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{y^2}{2\sigma_n^2}}. \quad (1.47)$$

The receiving-end decider determines the received signal and sets the decision threshold as b . If $y > b$, the signal is a “1”; if $y \leq b$, it is a “0”. Suppose the probability that the transmitted signal is “1” and it is erroneously received as “0” is $P(0/1)$, and the probability that the transmitted signal is “0” and it is erroneously received as “1” is $P(1/0)$; then, we have

$$P(0/1) = P(y \leq b) = \int_{-\infty}^b f_1(y) dy = 1 - \frac{1}{2} \operatorname{erfc}\left(\frac{b - \sqrt{P_r}}{\sqrt{2}\sigma_n}\right) \quad (1.48)$$

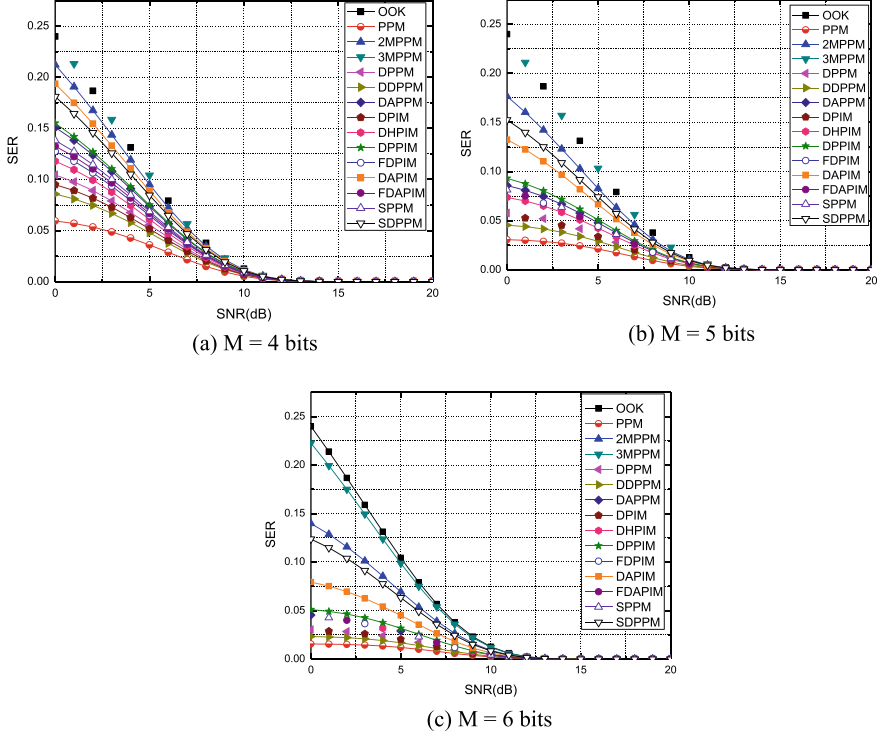


Fig. 1.9 Slot error rate of each modulation mode through a Gaussian channel [20]

$$P(1/0) = P(y > b) = \int_b^{\infty} f_0(y)dy = \frac{1}{2} \operatorname{erfc}\left(\frac{b}{\sqrt{2}\sigma_n}\right). \quad (1.49)$$

Assuming that the probability of sending a signal “1” is $P(1)$, and the probability of sending a signal “0” is $P(0)$, the total slot error rate of the system can be expressed as

$$P_{se} = P(1)P(0/1) + P(0)P(1/0). \quad (1.50)$$

In OOK modulation mode, $P(1) = P(0) = 1/2$; then, the slot error rate is

$$P_{se-OOK} = \frac{1}{2}P(0/1) + \frac{1}{2}P(1/0) = \frac{1}{4} \left[\left(2 - \operatorname{erfc}\left(\frac{b - \sqrt{P_r}}{\sqrt{2}\sigma_n}\right) \right) + \operatorname{erfc}\left(\frac{b}{\sqrt{2}\sigma_n}\right) \right]. \quad (1.51)$$

In PPM mode, $P(1) = 1/2^M$ and $P(0) = 2^M - 1/2^M$. Substituting into Eq. (1.50), the error slot rate of PPM can be obtained:

$$P_{se-PPM} = \frac{1}{2^{M+1}} \left[2 - \operatorname{erfc} \left(\frac{b - \sqrt{P_r}}{\sqrt{2}\sigma_n} \right) + (2^M - 1) \operatorname{erfc} \left(\frac{b}{\sqrt{2}\sigma_n} \right) \right]. \quad (1.52)$$

Let $l = 2^M - 1$; Eq. (1.52) can be written as

$$P_{se-PPM} = \frac{1}{2(l+1)} \left[2 - \operatorname{erfc} \left(\frac{b - \sqrt{P_r}}{\sqrt{2}\sigma_n} \right) + l \operatorname{erfc} \left(\frac{b}{\sqrt{2}\sigma_n} \right) \right]. \quad (1.53)$$

Similarly, the time-slot error-rate expressions of DPIM and DHPIM can be obtained as

$$P_{se-DPIM} = \frac{1}{2(l_{DPIM} + 1)} \left[2 - \operatorname{erfc} \left(\frac{b - \sqrt{P_r}}{\sqrt{2}\sigma_n} \right) + l_{DPIM} \operatorname{erfc} \left(\frac{b}{\sqrt{2}\sigma_n} \right) \right] \quad (1.54)$$

$$P_{se-DHPIM} = \frac{1}{2(l_{DHPIM} + 1)} \left[2 - \operatorname{erfc} \left(\frac{b - \sqrt{P_r}}{\sqrt{2}\sigma_n} \right) + l_{DHPIM} \operatorname{erfc} \left(\frac{b}{\sqrt{2}\sigma_n} \right) \right]. \quad (1.55)$$

$l_{DPIM} = (2^M + 1)/2$ and $l_{DHPIM} = (2^M + \alpha + 2)/3\alpha$. Therefore, the slot error rate of various modulation methods can be summarized as

$$P_{se} = \frac{1}{2(l+1)} \left[2 - \operatorname{erfc} \left(\frac{b - \sqrt{P_r}}{\sqrt{2}\sigma_n} \right) + l \operatorname{erfc} \left(\frac{b}{\sqrt{2}\sigma_n} \right) \right], \quad (1.56)$$

where l represents the ratio of the probability of “0” occurring to the probability of “1” occurring in the transmitted digital-signal sequence. The values of the different modulation methods are different, which is related to the number of modulation bits.

$$b = \frac{2\sigma_n^2 \ln(l) + P_r}{2\sqrt{P_r}} \quad (1.57)$$

We substitute Eq. (1.57) into Eq. (1.56) to obtain

$$P_{se} = \frac{1}{2(l+1)} \left[2 - \operatorname{erfc} \left(\frac{2\sigma_n^2 \ln(l) - P_r}{2\sqrt{2P_r}\sigma_n} \right) + l \operatorname{erfc} \left(\frac{2\sigma_n^2 \ln(l) + P_r}{2\sqrt{2P_r}\sigma_n} \right) \right]. \quad (1.58)$$

Assuming that the signal-to-noise ratio is represented by $\mu = \frac{P_r}{2\sigma_n^2}$, Eq. (1.58) can be written as

$$P_{se} = \frac{1}{2(l+1)} \left[2 - \operatorname{erfc} \left(\frac{\ln l - \mu}{2\sqrt{\mu}} \right) + l \cdot \operatorname{erfc} \left(\frac{\ln l + \mu}{2\sqrt{\mu}} \right) \right]. \quad (1.59)$$

The time-slot error rate of various modulation methods under a Gaussian channel is expressed in relation to the signal-to-noise ratio (SNR) as Eq. (1.59), and the expressions of l for different modulation methods are as follows:

$$l_{OOK} = 1,$$

$$l_{PPM} = 2^M - 1,$$

$$l_{DPPM} = (2^M - 1)/2,$$

$$l_{DPIM} = (2^M + 1)/2, \quad l_{DHPIM} = (2^M + \alpha + 2)/3\alpha,$$

$$l_{DPPIM} = (2^{M+1} + \alpha - 4)/(4 + 3\alpha), \quad l_{DDPPM} = (2^{M+1} + \alpha - 4)/3\alpha,$$

$$l_{FDPIIM} = (2^M + 1)/3, \quad l_{DAPIM} = (2^{M-1} + 2 - \beta)/(1 + \beta)$$

$$l_{FDAPIM} = (2^M + 2 - \beta)/(1 + \beta), \quad C_n^3 \geq 2^M$$

$$l_{SPPM} = (2^M - 1)/3, \quad l_{DAPPM} = (2^M - 1 - \beta)/(1 + \beta), \quad l_{3MPPM} = (n - 3)/3$$

$$l_{SDPPM} = (ns - 2)/2, \quad ns = \left(5 + \sqrt{25 - 4(6 - 2^{M+1})}\right)/2,$$

$$l_{2MPPM} = (nm - 2)/2, \quad nm = \left(1 + \sqrt{1 + 2^{M+3}}\right)/2$$

Following the numerical simulation of the slot error-rate expression, a comparison of the slot error rates of the different modulation methods under a Gaussian channel is shown in Fig. 1.9.

Figure 1.9 shows a comparison of the slot error rate of each modulation method under different modulation-bit numbers without considering the influence of turbulence. The involved pulse-width parameter was taken as 1, and the amplitude parameter was taken as 2. It can be seen from the three figures that the slot error rate of OOK modulation is unrelated to modulation order M . The larger the value of M , the smaller the slot error rate; when the value of M is constant, the slot error rate of OOK is the largest, and the slot error rate of PPM is the smallest. With the increase in M , the gap between the slot error rates of various modulation methods has widened.

1.2.2 Analysis of the Slot Error Rate Under a Turbulent Channel Model

(1) Average slot error rate

Assuming that the communication system has been synchronized, there is no inter-symbol crosstalk, and the channel-state information is known, the calculation expression of the time-slot error rate modulated by various modulation methods is shown in Eq. (1.58).

In a turbulent channel, after the receiving end receives the signal, the photodetector converts the optical signal into an electrical signal and then demodulates the original

signal. It can be known from photoelectric-detection theory that the relationship between the average optical power P_r and the light intensity h is $P_r = A_r h$, and A_r is the area of the receiving antenna [11]. Substituting this into Eq. (1.58), we obtain

$$P_{se} = \frac{1}{2(l+1)} \left[2 - \operatorname{erfc} \left(\frac{2\sigma_n^2 \ln l - A_r h}{2\sqrt{2\sigma_n^2 A_r h}} \right) + l \cdot \operatorname{erfc} \left(\frac{2\sigma_n^2 \ln l + A_r h}{2\sqrt{2\sigma_n^2 A_r h}} \right) \right]. \quad (1.61)$$

The current obtained by the optical signal received by the receiving end after photoelectric conversion is $I = \eta P_r = \eta A_r h$; then, the average power of the electrical signal is $P = I^2 R = \eta^2 A_r^2 h^2 R$, and the electrical-to-noise ratio is

$$\mu = \frac{P}{2\sigma_n^2} = \frac{\eta^2 A_r^2 h^2 R}{2\sigma_n^2} = \eta^2 A_r^2 R \frac{h^2}{2\sigma_n^2}. \quad (1.62)$$

Assuming that the receiving area is $A_r = 1$ and $\eta^2 A_r^2 R = 1$, the normalized average threshold-to-noise ratio (TNR) is $\mu_0 = h^2 / 2\sigma_n^2$, which can be obtained by substituting it into Eq. (1.61).

$$P_{se} = \frac{1}{2(l+1)} \left[2 - \operatorname{erfc} \left(\ln l \cdot \sqrt{\frac{h}{4\mu_0}} - \sqrt{\frac{\mu_0}{4h}} \right) + l \cdot \operatorname{erfc} \left(\ln l \cdot \sqrt{\frac{h}{4\mu_0}} + \sqrt{\frac{\mu_0}{4h}} \right) \right]. \quad (1.63)$$

A turbulent flow causes the laser intensity to constantly change; therefore, the value of the signal-to-noise ratio changes at all times, and only the average value of the slot error ratio can be obtained. The average slot error rate is expressed as [11]

$$\langle P_{se} \rangle = \int_0^\infty P_{se} \cdot f(h) dh. \quad (1.64)$$

The formula $f(h)$ is the probability-density function of channel state h in the case of turbulence; the expression form of $f(h)$ differs for different turbulence intensities.

(2) Average slot error rate in a weakly turbulent channel model

In the case of weak turbulence, the channel state follows a log-normal distribution, and its probability-density function is [12]

$$f_h(h) = \frac{1}{h\sigma\sqrt{2\pi}} \exp \left[-\frac{(\ln h + \sigma^2/2)^2}{2\sigma^2} \right], \quad (1.65)$$

where σ^2 is the variance of the log-normal distribution, which depends on the channel characteristics and is expressed as

$$\sigma^2 = \exp \left[\frac{0.49\delta^2}{(1 + 0.18d^2 + 0.56\delta^{12/5})^{7/6}} + \frac{0.51\delta^2}{(1 + 0.9d^2 + 0.62d^2\delta^{12/5})^{5/6}} \right] - 1. \quad (1.66)$$

$d = \sqrt{kD^2/4L}$, $k = 2\pi/\lambda$ is the light-wave number, $\delta^2 = 1.23C_n^2k^{7/6}L^{11/6}$ is the Rytov variance, and C_n^2 is the atmospheric refractive-index structure constant, which depends on the altitude. The peripheral model of C_n^2 typically uses the Hufnagel–Valley model, described as [12]

$$C_n^2(h) = 0.00594(v/27)^2(10^{-5}h)^{10}e^{h/1000} + 2.7 \times 10^{-6}e^{-h/1500} + Ae^{-h/1000}, \quad (1.67)$$

where h is the altitude (m), v is the wind speed (m/s), and A is the value of $C_n^2(0)$. For a free-space optical (FSO) communication link near the ground, the value of C_n^2 varies from $10^{-17}m^{-2/3}$ to $10^{-13}m^{-2/3}$ in the process of changing from turbulent to strong turbulence. The average value is generally $10^{-15}m^{-2/3}$.

From Eqs. (1.63), (1.64), and (1.65), the average slot error rate in the case of weak turbulence is

$$\langle P_{se} \rangle = \frac{1}{2\sqrt{2\pi}\sigma(l+1)} \int_0^\infty [2 - \operatorname{erfc}(d) + l \cdot \operatorname{erfc}(k)]h^{-1} \exp\left(-\frac{(\ln h + \sigma^2)^2}{2\sigma^2}\right)dh \quad (1.68)$$

where d and k are denoted as

$$d = \ln l \cdot \sqrt{\frac{h}{4\mu_0}} - \sqrt{\frac{\mu_0}{4h}} \quad (1.69)$$

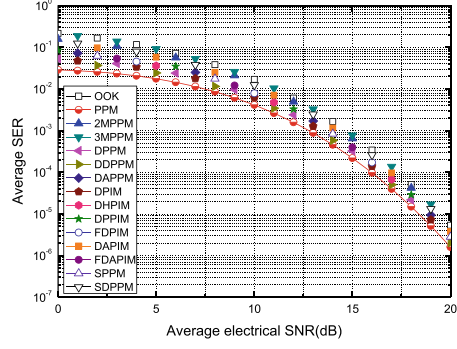
$$k = \ln l \cdot \sqrt{\frac{h}{4\mu_0}} + \sqrt{\frac{\mu_0}{4h}}. \quad (1.70)$$

Owing to the different modulation modes, l of each modulation mode can be determined using Eq. (1.59), the light wavelength λ is 850 nm, and the receiving-end aperture D is 8 cm. A comparison of the average slot error rates of different modulation methods under weak turbulence is shown in Fig. 1.10.

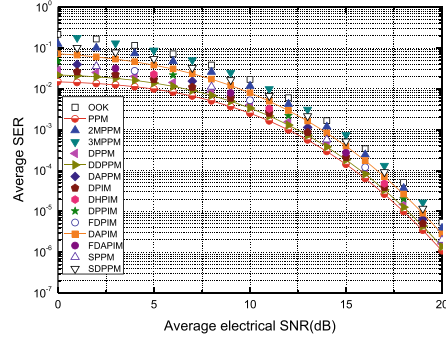
The three sub-figures in Fig. 1.10 show comparisons of the average slot error rates of different modulation methods under different transmission distances and different modulation-bit numbers in the case of weak turbulence. From these three figures, it can be seen that, for various modulations, the average time-slot error rate of the method decreases with an increase in the number of modulation bits; it also increases with an increase in the transmission distance.

Figure 1.10a,b compare the average slot error rate of each modulation method at the same distance and different modulation-bit numbers. When the number of

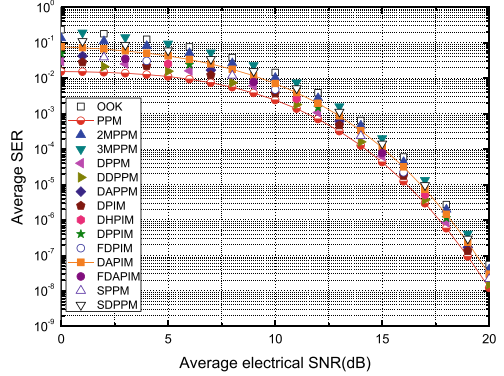
Fig. 1.10 Average slot error rate of each modulation method in a weakly turbulent channel [20]



(a) $L = 5000m, M = 5bit, C_n^2 = 1 \times 10^{-15} m^{-2/3}$



(b) $L = 5000m, M = 6bit, C_n^2 = 1 \times 10^{-15} m^{-2/3}$



(c) $L = 3000m, M = 6bit, C_n^2 = 1 \times 10^{-15} m^{-2/3}$

transmission bits is five, the average slot error rates of 2MPPM and 3PPM modulation schemes with OOK and pulse numbers of 2 and 3 are the highest and similar, followed by SDPPM and DAPIM, then DPPIM, DAPPM, SPPM, FDAPIM, FDPIM, DHPIM, DPPM, DPIM, and DDPPM. The average time-slot error rates of these eight modulation methods are very similar, with the smallest being PPM.

When the number of transmission bits is six, the difference in the average slot error rate between the modulation methods increases. The largest is still OOK, followed by 3MPPM, 2MPPM, SDPPM, and DAPIM. Following are DPPIM, DAPPM, SPPM, FDAPIM, FDPIM and DHPIM, which are very similar, followed by DPPM, DPIM, DDPPM, and finally PPM.

It can be seen that for modulation methods with similar average symbol lengths, the average time-slot error rate of a modulation method with a large number of pulses is higher; for a modulation method with the same number of pulses, the average time-slot error rate of a larger average symbol length is slightly higher. When the number of modulation bits increases from five to six, the average slot error rate decreases from $10^{-5} \sim 10^{-6}$ to 10^{-6} .

Figure 1.10b,c compare the average time-slot error rate of various modulation methods with the same modulation-bit number and different transmission distances. From these two figures, it can be seen that when the transmission distance changes from 5000 to 3000 m, the average time-slot error rate is reduced from 10^{-6} to 10^{-8} , and the order of the average time-slot error rate of each modulation mode does not change; however, the gap between them decreases.

(3) Average slot error rate in a moderately strong turbulent channel model

As the atmospheric-turbulence intensity changes from weak to strong, the state h of the channel obeys the gamma-gamma distribution, and its probability density function is [12]

$$f_h(h) = \frac{2(\alpha\beta)^{(\alpha+\beta)/2}}{\Gamma(\alpha)\Gamma(\beta)} \cdot h^{\frac{\alpha+\beta}{2}-1} \cdot K_{\alpha-\beta}(2\sqrt{\alpha\beta}h), \quad (1.71)$$

where $\Gamma(\cdot)$ is the gamma function and K_v is the modified Bessel function of the second order v . α and β are the parameters of the flicker index, and their expressions in the case of plane waves are

$$\alpha = \left\{ \exp \left[\frac{0.49\sigma_0^2}{(1 + 0.18d^2 + 0.56\sigma_0^{12/5})^{7/6}} \right] - 1 \right\}^{-1} \quad (1.72)$$

$$\beta = \left\{ \exp \left[\frac{0.51\sigma_0^2}{(1 + 0.9d^2 + 0.62d^2\sigma_0^{12/5})^{7/6}} \right] - 1 \right\}^{-1}. \quad (1.73)$$

From Eqs. (1.63), (1.64), and (1.71), the average slot error rate under moderately strong turbulence can be obtained as

$$\langle P_{se} \rangle = \frac{(\alpha\beta)^{(\alpha+\beta)/2}}{\Gamma(\alpha)\Gamma(\beta)(l+1)} \int_0^\infty [2 - \operatorname{erfc}(d) + l \cdot \operatorname{erfc}(k)] h^{\frac{\alpha+\beta}{2}-1} K_{\alpha-\beta}(2\sqrt{\alpha\beta h}) dh, \quad (1.74)$$

where d and k are represented by Eqs. (1.69) and (1.70).

Figure 1.11 shows a comparison of the average slot error rate of each modulation method under moderate to strong turbulence, according to the calculation expression.

Figure 1.11 shows a comparison of the slot error rate of each modulation method under different transmission distances and different modulation-bit numbers, in the case of moderate to strong turbulence. From the three subfigures, it can be seen that the variation trends of various modulation methods under the corresponding conditions are consistent with those of weak turbulence.

When the modulation-bit number increases from five bits to six bits, the average time-slot error rate decreases from 10^{-5} to $10^{-5} \sim 10^{-6}$; when the modulation bit number is the same, the average time-slot error rate decreases from 5000 to 3000 m when the transmission distance is reduced from $10^{-5} \sim 10^{-6}$ to 10^{-6} . Compared with the weak-turbulence case, the average slot error rate is larger in the case of moderately strong turbulence.

(4) Average slot error rate under a strongly turbulent channel model

In the case of strong turbulence, the light intensity obeys the K distribution, and the probability-density function is [13]

$$f_h(h) = \frac{2a^{(a+1)/2}}{\Gamma(a)} \cdot h^{(a-1)/2} K_{a-1}(2\sqrt{ah}), \quad (1.75)$$

where Γ is the gamma function, K_ν is the modified Bessel function of the second order, and a is a channel parameter related to the effective number of discrete scatterers. In the strong-turbulence range, the turbulence intensity increases with an decrease in a .

Combining Eqs. (1.63), (1.64), and (1.75), the average slot error rate in the case of strong turbulence can be obtained as follows:

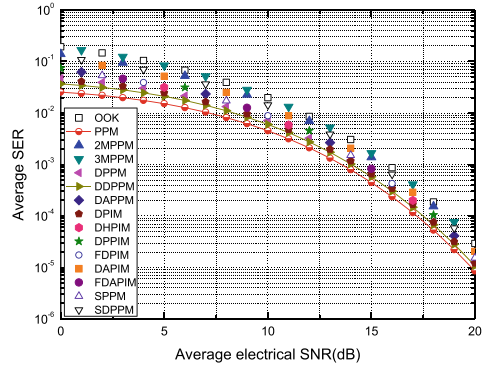
$$\langle P_{se} \rangle = \frac{a^{(a+1)/2}}{\Gamma(a)(l+1)} \int_0^\infty [2 - \operatorname{erfc}(d) + l \cdot \operatorname{erfc}(k)] h^{\frac{a-1}{2}} K_{a-1}(2\sqrt{ah}) dh, \quad (1.76)$$

where d and k are represented by Eqs. (1.69) and (1.70), respectively.

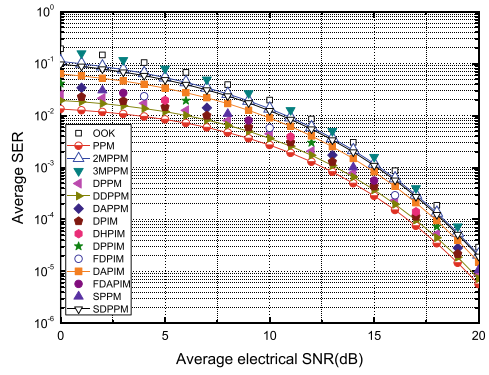
Figure 1.12 shows a comparison of the average slot error rate of each modulation method in the case of strong turbulence, according to the calculation expression.

Figure 1.12 shows a comparison of the average slot error rate of different modulation modes under the strong-turbulence condition with different a values and modulation bit numbers. It can be seen from the subfigures that, with the change in the

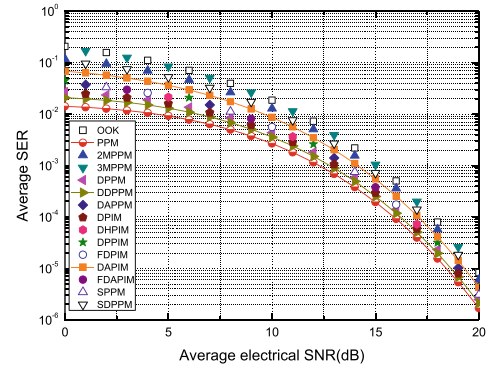
Fig. 1.11 Average time-slot error rate of each modulation mode under a moderately strong turbulent channel [20]



(a) $L = 5000m, M = 5bit, C_n^2 = 9 \times 10^{-15} m^{-2/3}$



(b) $L = 5000m, M = 6bit, C_n^2 = 9 \times 10^{-15} m^{-2/3}$



(c) $L = 3000m, M = 6bit, C_n^2 = 9 \times 10^{-15} m^{-2/3}$

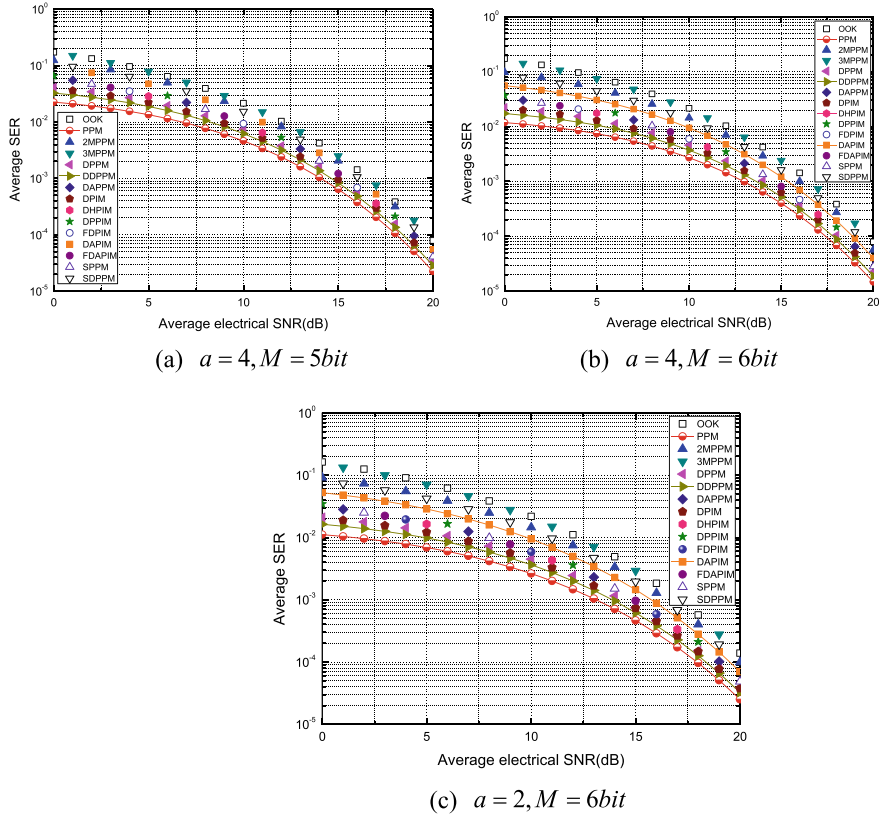


Fig. 1.12 Average time-slot error rate of each modulation method in a strong turbulent channel [20]

values of a and M , the size order of the above modulation modes does not change. When a is constant, the average slot error rate of each modulation mode increases with the number of modulation bits M . When the value of M is constant, the average time-slot error rate of each modulation method decreases with an increase in a . The average time-slot error rate in the three figures is 10^{-4} to 10^{-5} , which is higher than that of the medium-strong turbulent flow. The average time-slot error rate is larger.

1.3 Channel Capacity

The channel capacity refers to the maximum average information rate that a channel can transmit. This section analyzes the transmission capacities of various modulation modes and the average channel capacities of different modulation modes under turbulent channels.

1.3.1 Transmission Capacity

The transmission capacity refers to the amount of information that can be transmitted per unit of time with the same data-transmission rate. The pseudo data-transmission rate is R_b bits/s. In the OOK modulation mode, the time-slot width is $T_b = 1/R_b$.

If each symbol corresponds to M binary information bits, with the same information bit rate, the time-slot width of the PPM modulation mode is $T_s = MT_b/2^M$, and the transmission capacity of PPM is

$$C_{PPM} = \frac{M}{T_s L_{PPM}} = R_b. \quad (1.77)$$

With the same time-slot width, the symbol length of DPPM is changed, compared with that of PPM, and the average symbol length is shortened, resulting in a higher transmission capacity. The average symbol length of DPPM is $L_{DPPM} = (2^M + 1)/2$, and the transmission capacity is

$$C_{DPPM} = \frac{M}{T_s L_{DPPM}} = \frac{2^{M+1} R_b}{2^M + 1}. \quad (1.78)$$

Similarly, according to the average symbol length of various modulation methods mentioned in Sect. 1.2, the transmission capacities of DPIM, DHPIM, and other modulation methods can be obtained as

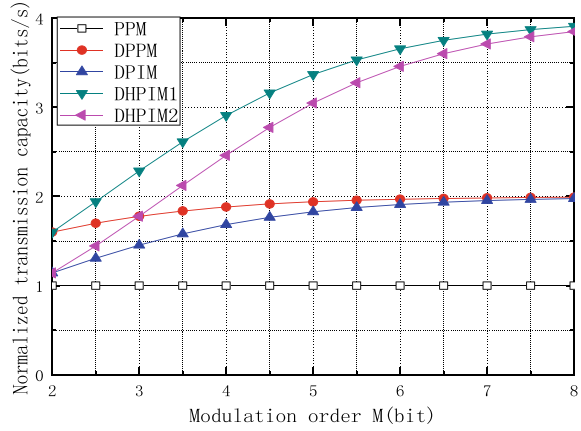
$$C_{DPIM} = \frac{M}{T_s L_{DPIM}} = \frac{2^{M+1} R_b}{2^M + 3} \quad (1.79)$$

$$C_{DHPIM} = \frac{M}{T_s L_{DHPIM}} = \frac{2^{M+1} R_b}{2^{M-1} + 2\alpha + 1}. \quad (1.80)$$

A numerical-simulation comparison of the transmission capacities of the above-mentioned modulation methods was carried out, and a simulation comparison based on OOK is shown in Fig. 1.13.

Figure 1.13 shows a comparison of the transmission capacities of the different modulation schemes. It can be observed from the figure that the transmission capacity of each modulation method increases logarithmically with an increase in the number of modulation bits. The transmission capacity of DHPIM is the largest, approaching four bits/s, followed by PPM and DPIM. The transmission capacity is close to two bits/s, and the smallest is PPM. For modulation modes with changing pulse-width parameters, the transmission capacity decreases with an increase in the pulse-width parameters.

Fig. 1.13 Normalized transmission capacities of various modulation methods [20]



1.3.2 Average Channel Capacity with Turbulent Channels

(1) Average channel capacity

In a turbulent atmospheric channel, assuming that the channel is memoryless, stationary, and ergodic, the statistical channel model can be expressed as [14]

$$y = sx + n = \eta Ix + n, \quad (1.81)$$

where y represents the signal at the receiving end, x represents the modulated signal, I represents the instantaneous light intensity of the received signal, η represents the photoelectric-conversion efficiency of the receiving end, and n is Gaussian white noise with a mean value of 0 and a variance of $N_0/2$, which is produced by an electronic device. η can be expressed as [14]

$$\eta = \frac{ve\lambda}{h_p c}, \quad (1.82)$$

where v is the quantum efficiency, e is the electron charge, λ is the signal wavelength, h_p is Planck's constant, and c is the speed of light.

The wireless-optical channel is a random time-varying channel, and the signal-to-noise ratio of the instantaneous electrical signal received by the receiver is also a random variable, expressed as [15]

$$\gamma = \frac{s^2}{N_0} = \frac{\eta^2 I^2}{N_0}. \quad (1.83)$$

The expression of the average channel capacity with respect to the signal-to-noise ratio in a turbulent channel is [16]

$$\langle C \rangle = \int_0^\infty B \log_2(1 + \gamma) p_\gamma(\gamma) d\gamma, \quad (1.84)$$

where $p_\gamma(\gamma)$ is the probability-density function of the signal-to-noise ratio γ , which varies according to the distribution that the light intensity I obeys.

(2) Turbulent channel model

a. Log-normal distribution model

Under a weak turbulent channel, the light intensity I follows a log-normal distribution, and its probability-density function is [12]

$$f_I(I) = \frac{1}{I\sigma\sqrt{2\pi}} \exp\left[-\frac{(\ln I + \sigma^2/2)^2}{2\sigma^2}\right], \quad (1.85)$$

where σ^2 is the variance of the log-normal distribution, which depends on the channel characteristics, and is expressed as in Eq. (1.66).

The probability-density function of the signal-to-noise ratio in the case of weak turbulence can be obtained using the following formula:

$$p_\gamma(\gamma) = \frac{1}{2\gamma\sigma\sqrt{2\pi}} \exp\left(-\frac{(\ln(\gamma/\mu) + \sigma^2)^2}{8\sigma^2}\right). \quad (1.86)$$

$\mu = \frac{(\eta E[I])^2}{N_0} = \frac{\eta^2}{N_0}$ is the average signal-to-noise ratio after normalizing the light intensity.

Substituting Eq. (1.86) into Eq. (1.84), the expression for the average channel capacity with respect to the signal-to-noise ratio is

$$\langle C \rangle = \frac{B}{2 \ln(2)\sigma\sqrt{2\pi}} \int_0^\infty \frac{\ln(1 + \gamma)}{\gamma} \exp\left(-\frac{(\ln(\gamma) - \ln(\mu) + \sigma^2)^2}{8\sigma^2}\right) d\gamma. \quad (1.87)$$

Let $u = \ln(\gamma)$ and $\xi = \ln(\mu) - \sigma^2$. Equation (1.85) can be expressed as

$$\langle C \rangle = \frac{B}{2 \ln(2)\sigma\sqrt{2\pi}} \left[\int_0^\infty \ln(1 + e^{-u}) e^{-\frac{(u+\xi)^2}{8\sigma^2}} du + \int_0^\infty \ln(1 + e^u) e^{-\frac{(u-\xi)^2}{8\sigma^2}} du \right]. \quad (1.88)$$

Because $\ln(1 + e^u) = u + \ln(1 + e^{-u})$, we substitute into the above formula to obtain

$$\int_0^\infty u e^{-\frac{(u-\xi)^2}{8\sigma^2}} du = 4\sigma^2 e^{-\frac{\xi^2}{8\sigma^2}} + \xi \sqrt{2\pi}\sigma \operatorname{erfc}\left(-\frac{\xi}{2\sqrt{2}\sigma}\right). \quad (1.90)$$

For $0 \leq x \leq 1$,

$$\ln(1+x) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} x^k. \quad (1.91)$$

For $0 \leq e^{-u} \leq 1$ and $u \geq 0$,

$$\ln(1+e^{-u}) = \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} e^{-ku}. \quad (1.92)$$

Then, the average channel capacity under a weakly turbulent channel can be expressed as

$$\begin{aligned} \langle C \rangle = & \frac{B}{2 \ln(2) \sigma \sqrt{2\pi}} \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \left(\int_0^{\infty} e^{-ku} e^{-\frac{(u+\xi)^2}{8\sigma^2}} du + \int_0^{\infty} e^{-ku} e^{-\frac{(u-\xi)^2}{8\sigma^2}} du \right) \\ & + \frac{2\sigma B}{\ln(2) \sqrt{2\pi}} e^{-\frac{\xi^2}{8\sigma^2}} + \frac{\xi B}{2 \ln(2)} \operatorname{erfc} \left(-\frac{\xi}{2\sqrt{2}\sigma} \right). \end{aligned} \quad (1.93)$$

Because $\int_0^{\infty} e^{-ku} e^{-\frac{(u\pm\xi)^2}{8\sigma^2}} du = e^{-\frac{\xi^2}{8\sigma^2}} \sigma \sqrt{2\pi} \operatorname{erfcx} \left(\sqrt{2}\sigma k \pm \frac{\xi}{2\sqrt{2}\sigma} \right)$, the closed expression of the average channel capacity with respect to the average signal-to-noise ratio is

$$\begin{aligned} \langle C \rangle = & \frac{B}{2 \ln(2)} e^{-\frac{\xi^2}{8\sigma^2}} \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} \left[\operatorname{erfcx} \left(\sqrt{2}\sigma k + \frac{\xi}{2\sqrt{2}\sigma} \right) + \operatorname{erfcx} \left(\sqrt{2}\sigma k - \frac{\xi}{2\sqrt{2}\sigma} \right) \right] \\ & + \frac{2\sigma B}{\ln(2) \sqrt{2\pi}} e^{-\frac{\xi^2}{8\sigma^2}} + \frac{\xi B}{2 \ln(2)} \operatorname{erfc} \left(-\frac{\xi}{2\sqrt{2}\sigma} \right) \end{aligned} \quad (1.94)$$

b. Gamma–gamma distribution model

In a medium-strong turbulent environment, that is, in the process of changing the turbulence intensity from weak to strong, the light intensity I obeys the gamma–gamma distribution, and the expression of the probability-density function is [12]

$$f_I(I) = \frac{2(\alpha\beta)^{(\alpha+\beta)/2}}{\Gamma(\alpha)\Gamma(\beta)} \cdot I^{\frac{\alpha+\beta}{2}-1} \cdot K_{\alpha-\beta}(2\sqrt{\alpha\beta}I). \quad (1.95)$$

Among them, α and β are the parameters of the flicker index, and their expressions in the case of plane waves are given by Eqs. (1.72) and (1.73), respectively.

According to the formula, the probability-density function of the signal-to-noise ratio γ under the gamma–gamma distribution is obtained as

$$p_{\gamma}(\gamma) = \frac{(\alpha\beta)^{(\alpha+\beta)/2}}{\Gamma(\alpha)\Gamma(\beta)} \cdot \frac{\gamma^{(\alpha+\beta)/4-1}}{\mu^{(\alpha+\beta)/4}} K_{\alpha-\beta} \left(2\sqrt{\alpha\beta} \sqrt{\frac{\gamma}{\mu}} \right). \quad (1.96)$$

Substituting Eq. (1.96) into Eq. (1.84), the expression of the average channel capacity under the gamma–gamma distribution is

$$\langle C \rangle = \frac{B(\alpha\beta)^{(\alpha+\beta)/2}}{\ln 2 \cdot \Gamma(\alpha)\Gamma(\beta)\mu^{(\alpha+\beta)/4}} \int_0^\infty \gamma^{(\alpha+\beta)/4-1} \cdot \ln(1+\gamma) K_{\alpha-\beta} \left(2\sqrt{\alpha\beta} \sqrt{\frac{\gamma}{\mu}} \right) d\gamma. \quad (1.97)$$

Because $\ln(1+x)$ and $K_\nu(\cdot)$, they can be expressed by the Meijer-G function as [17]

$$\ln(1+x) = G_{2,2}^{1,2} \left(x \middle|_{1,0}^{1,1} \right) \quad (1.98)$$

$$K_\nu(\sqrt{z}) = \frac{1}{2} G_{0,2}^{2,0} \left(\frac{z}{4} \middle|_{v/2, -v/2}^- \right). \quad (1.99)$$

Then, according to the properties of the Meijer-G function [18],

$$\begin{aligned} & \int_0^\infty \tau^{\alpha-1} G_{u,v}^{s,t} \left(\sigma \tau \middle|_{d_1, \dots, d_s, d_{s+1}, \dots, d_v}^{c_1, \dots, c_t, c_{t+1}, \dots, c_u} \right) G_{p,q}^{m,n} \left(\omega \tau^{l/k} \middle|_{b_1, \dots, b_m, b_{m+1}, \dots, b_q}^{a_1, \dots, a_n, a_{n+1}, \dots, a_p} \right) d\tau \\ &= \frac{k^\mu l^{(v-u)\alpha+\rho-1}}{(2\pi)^{(l-1)b^*+(k-1)c^*}} \\ & \sigma^{-\alpha} G_{kp+lv, kq+lu}^{km+lt, kn+ls} \left(\frac{\omega^k k^{k(p-q)}}{\sigma^l l^{l(u-v)}} \middle|_{\frac{b_1}{k}, \dots, \frac{b_1+k-1}{k}, \dots, \frac{b_m}{k}, \dots, \frac{b_m+k-1}{k}, \frac{1-\alpha-c_1}{l}, \dots, \frac{l-\alpha-c_1}{l}, \dots, \frac{1-\alpha-c_l}{l}, \dots, \frac{l-\alpha-c_l}{l}}^{\frac{a_1}{k}, \dots, \frac{a_1+k-1}{k}, \dots, \frac{a_n}{k}, \dots, \frac{a_n+k-1}{k}, \frac{1-\alpha-d_1}{l}, \dots, \frac{l-\alpha-d_1}{l}, \dots, \frac{1-\alpha-d_s}{l}, \dots, \frac{l-\alpha-d_s}{l}} \right). \end{aligned} \quad (1.100)$$

Combining Eqs. (1.97)–(1.100), the closed expression of the average channel capacity is obtained as

$$\begin{aligned} \langle C \rangle &= \frac{B(\alpha\beta)^{(\alpha+\beta)/2}}{4\pi \cdot \ln 2 \cdot \Gamma(\alpha)\Gamma(\beta)\mu^{(\alpha+\beta)/4}} \\ & G_{2,6}^{6,1} \left(\frac{(\alpha\beta)^2}{16\mu} \middle|_{\frac{\alpha-\beta}{4}, \frac{\alpha-\beta+2}{4}, \frac{\beta-\alpha}{4}, \frac{\beta-\alpha+2}{4}, \frac{-\alpha-\beta}{4}, \frac{-\alpha-\beta}{4}}^{-\frac{\alpha+\beta}{4}, -\frac{\alpha+\beta}{4}+1} \right). \end{aligned} \quad (1.101)$$

c. *K*-distribution model

In the case of strong turbulence, the light intensity I obeys the *K* distribution, and the probability-density function is [13]

$$f_I(I) = \frac{2a^{(a+1)/2}}{\Gamma(a)} \cdot I^{(a-1)/2} K_{a-1}(2\sqrt{aI}), \quad (1.102)$$

where a represents the channel parameter related to the number of discrete scatterers.

According to the formula, the probability-density function of the random variable function is calculated, and the probability-density function of the signal-to-noise ratio γ under the K distribution is

$$p_\gamma(\gamma) = \frac{a^{(a+1)/2}}{\Gamma(a)} \cdot \frac{\gamma^{(a-3)/4}}{\mu^{(a+1)/4}} K_{a-1} \left(2\sqrt{a\sqrt{\frac{\gamma}{\mu}}} \right). \quad (1.103)$$

Substituting Eq. (1.103) into Eq. (1.85), we obtain the closed expression of the average channel capacity under the K distribution as

$$\langle C \rangle = \frac{B \cdot (a/\mu)^{(a+1)/2}}{4\pi \Gamma(a) \ln 2} G_{2,6}^{6,1} \left(\frac{a^2}{16\mu} \left| \begin{matrix} -\frac{a+1}{4}, \frac{3-a}{4} \\ \frac{a-1}{4}, \frac{a+1}{4}, \frac{1-a}{4}, \frac{3-a}{4}, \frac{a+1}{4}, -\frac{a+1}{4} \end{matrix} \right. \right). \quad (1.104)$$

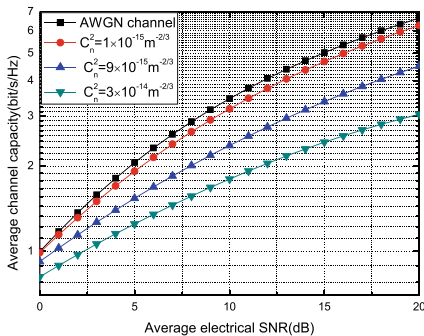
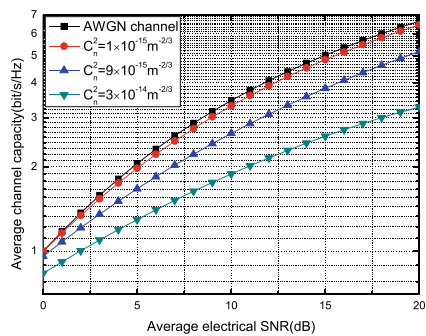
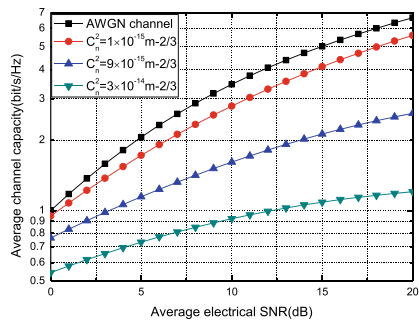
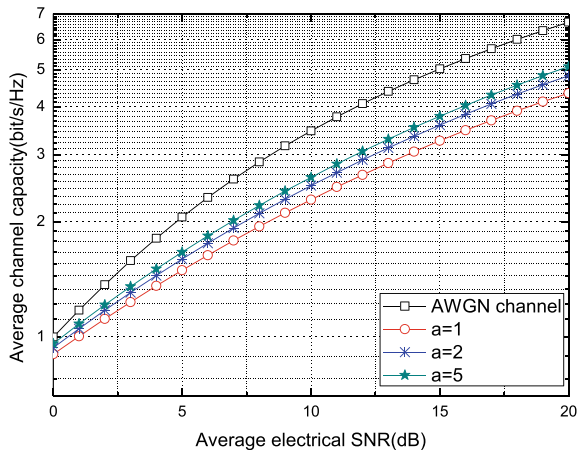
It can be seen from the mathematical expression of the above-average channel capacity that the channel capacity of an optical wireless communication system depends on the wavelength λ of the laser, the transmission distance L , and the diameter D of the telescope of the receiving detector. The intensity of the turbulent flow is characterized by the atmospheric refractive-index structure constant C_n^2 .

In the case of strong turbulence, channel parameter a is used to characterize the turbulence intensity when the light intensity obeys the K distribution. Reference [16] numerically simulated the expression of the average channel capacity on the signal-to-noise ratio under the above three turbulent channels, where the wavelength l is 850 nm and 1550 nm, the transmission distance L is 3000 m and 5000 m, and the receiver aperture D is taken as 2 cm. The values of C_n^2 are $1 \times 10^{-15} m^{-2/3}$, $9 \times 10^{-15} m^{-2/3}$, and $3 \times 10^{-14} m^{-2/3}$, in the case of weak, medium, and strong turbulence, respectively.

In the case of strong turbulence, the channel parameters when the light intensity obeys the K distribution are 1, 2, and 5. A comparison of the average channel capacity with respect to the signal-to-noise ratio under different turbulent channels is shown in Figs. 1.14 and 1.15.

Figure 1.14 shows a comparison of the average channel capacity at different transmission distances and optical wavelengths under turbulent channels. Medium-strong and strong turbulence channels are both simulated by the gamma-gamma distribution model and are distinguished by different C values. It can be observed from these three figures that the average channel capacity increases with an increase in the signal-to-noise ratio, decreases with an increase in the transmission distance, and increases with an increase in the optical wavelength.

Figure 1.15 shows a comparison of the average channel capacity when the light intensity follows a K distribution under strong turbulence. As can be seen from the figure, the average channel capacity increases with an increase in the signal-to-noise ratio and increases with an increase in channel parameter a . This is consistent with the decrease in turbulence intensity with increasing a [13].

(a) $L = 3000m, \lambda = 850nm$ (b) $L = 3000m, \lambda = 1550nm$ (c) $L = 5000m, \lambda = 850nm$ **Fig. 1.14** Average channel capacity under a turbulent channel [20]**Fig. 1.15** Average channel capacity when the light intensity follows the K distribution in a strongly turbulent channel [20]

d. Average channel capacities of the different modulation methods

There are many modulation modes in optical wireless communication systems, and the channel capacity of each mode is different. The previous section summarized the average channel capacity under a turbulent channel and performed a simulation analysis.

From the mathematical formula of the average channel capacity, it can be observed that the key to distinguishing the average channel capacity of different modulation methods lies in the signal-to-noise ratio. The representation of the noise ratio does not distinguish between the different modulations; therefore, it is necessary to change its representation. The optical intensity in Eq. (1.80) is changed to the optical power to represent the change in the laser. The average optical power of the signal at the receiving end is expressed as [19]

$$P_R(h) = P_T \left(\frac{\eta A}{\lambda L} \right)^2 h, \quad (1.105)$$

where h is the channel state, P_T is the average power of the signal at the transmitter, η is the photon efficiency of the transmitter and receiver, A is the area of the telescope, $A = \pi D^2/4$, D is the diameter of the telescope, and L is the distance between the transmitter and receiver.

According to photoelectric-detection theory, the photocurrent detected on the detector at the receiving end is $I = \eta_z P$, which is the photoelectric conversion efficiency, η_z ; therefore, the electrical signal power at the receiving end is p , and the electrical-to-noise ratio at the receiving end is

$$\gamma(h) = \frac{P^2}{2\sigma_n^2} = \frac{\eta_z^2 P_T^2 R}{2\sigma_n^2} \left(\frac{\eta A}{\lambda L} \right)^4 h^2. \quad (1.106)$$

According to the relationship between the average power and the peak power of the transmitter mentioned in Sect. 1.2, the relationship between the TNR and the peak power of the transmitter can be obtained; thus, the average channel capacity of different modulation methods can be expressed by this relationship. The calculation for the average channel capacity can be expressed as

$$\langle C \rangle = \int_0^\infty B \log_2(1 + \gamma(h)) f_h(h) dh. \quad (1.107)$$

In this formula, the signal-to-noise ratio is expressed as Eq. (1.105).

In the case of weak turbulence, the channel state h obeys a log-normal distribution, and the TNR at the receiving end is related to the average power at the transmitting end. According to the relationship between the average transmitting power and the peak power, Eqs. (1.105), (1.106), and (1.81), the average channel capacity of different modulation modes with respect to the peak transmit power P_e is expressed as

$$\langle C \rangle = \frac{B}{\ln 2 \cdot \sigma \sqrt{2\pi}} \int_0^\infty \ln \left(1 + \frac{\eta_z^2 t^2 P_c^2 R}{2\sigma_n^2} \left(\frac{\eta A}{\lambda L} \right)^4 h^2 \right) \cdot h^{-1} \exp \left[-\frac{(\ln h + \sigma^2/2)^2}{2\sigma^2} \right] h. \quad (1.108)$$

In the formula, t represents the ratio of the average transmit power to the peak transmit power, and its value differs with the different modulation methods:

$$t_{OOK} = 1/2,$$

$$t_{DAPPM} = (1 + \beta)/2^M,$$

$$t_{SDPPM} = 2/ns,$$

$$t_{2MPPM} = 2/nm,$$

$$t_{3MPPM} = 3/n, \quad ns = \left(5 + \sqrt{25 - 4(6 - 2^{M+1})} \right)/2, \quad nm = \left(1 + \sqrt{1 + 2^{M+3}} \right)/2$$

$$C_n^3 \geq 2^M,$$

$$t_{DPIM} = 2/(2^M + 3), \quad t_{DHPIM} = 3\alpha/(2^M + 4\alpha + 2),$$

$$t_{FDPIIM} = 3/(2^M + 4), \quad t_{DPPIM} = (4 + 3\alpha)/(2^{M+1} + 4\alpha),$$

$$t_{DAPIM} = (1 + \beta)/(2^{M-1} + 3), \quad t_{FDAPIM} = (1 + \beta)/(2^M + 3)$$

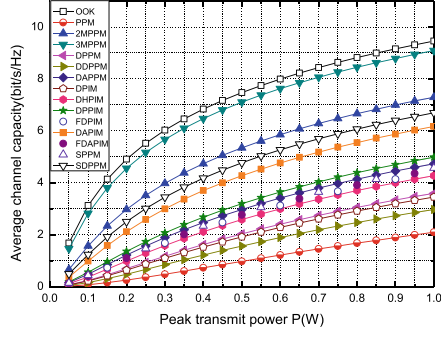
$$t_{SPPM} = 3/(2^M + 2) \text{ and } t_{PPM} = 1/2^M.$$

Equation (1.107) was numerically simulated. The photoelectric conversion efficiency η_z was 0.5, the photon efficiency η was 0.8, the receiving-end aperture D was 8 cm, the wavelength was 850 nm, and the noise variance was 2.5×10^{-5} A. A comparison of the average channel capacity of the various modulation methods in the case of weak turbulence is shown in Fig. 1.16.

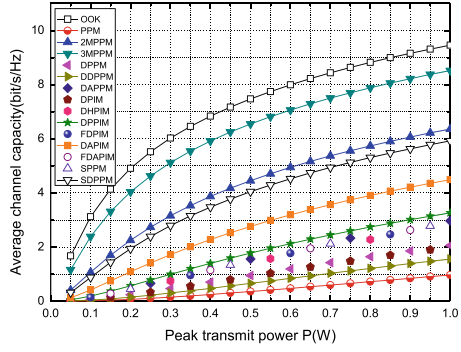
Figure 1.16 shows a comparison of the average channel capacity of different modulation methods at different transmission distances and different modulation-bit numbers in the case of weak turbulence. It can be seen from the figure that the average channel capacity increases with the increase in the peak-transmit power, decreases with the increase in the transmission distance, and decreases with an increase in the number of modulation bits. The average channel capacity of OOK is the largest and PPM is the smallest. The order of the average channel capacities of the various modulation modes is OOK, 3MPPM, 2MPPM, SDPPM, DAPIM, DPPIM, DAPPM, SPPM, FDAPIM, FDPIM, DHPIM, DPPM, DPIM, DDPPM, and PPM.

Figure 1.16a, b show the comparison of the average channel capacity of different modulation methods when the numbers of modulation bits are five and six and the distance is 5000 m. It can be seen from these two figures that the average channel

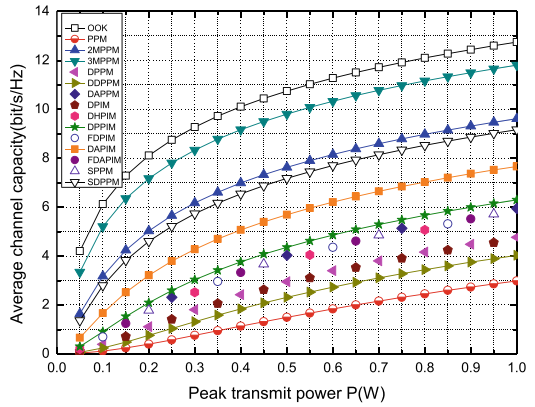
Fig. 1.16 Average channel capacity of each modulation mode under a weak turbulence channel [20]



(a) $L = 5000m, M = 5bit, C_n^2 = 1 \times 10^{-15} m^{-2/3}$



(b) $L = 5000m, M = 6bit, C_n^2 = 1 \times 10^{-15} m^{-2/3}$



(c) $L = 3000m, M = 6bit, C_n^2 = 1 \times 10^{-15} m^{-2/3}$

capacity of OOK is the same, regardless of the number of modulation bits. The average channel capacity of the other modulation methods when the number of modulation bits is five is higher than that when the number of modulation bits is six (1–2 bits/s/Hz).

Figure 1.16b, c show the comparison of the average channel capacity of different modulation methods when the number of modulation bits is six and the transmission distance is 3000 m and 5000 m, respectively. It can be observed from these two figures that the average channel capacity when the transmission distance is 3000 m is higher than that when the transmission distance is 5000 m; specifically, greater than 3–4 bits/s/Hz. Under these three conditions, the maximum channel capacity reached nearly 13 bits/s/Hz.

In the case of moderate-to-strong turbulence, the channel state h obeys the gamma-gamma distribution. According to the relationship between the signal-to-noise ratio and the average power of the transmitter, the expression of the peak-transmit power of the average channel capacity of each modulation method, in the case of moderate-to-strong turbulence, is as follows:

$$\langle C \rangle = \frac{2B(\alpha\beta)^{(\alpha+\beta)/2}}{\ln 2 \cdot \Gamma(\alpha)\Gamma(\beta)} \int_0^\infty \ln \left(1 + \frac{\eta_z^2 t^2 P_c^2 R}{2\sigma_n^2} \left(\frac{\eta A}{\lambda L} \right)^4 h^2 \right) \cdot h^{\frac{\alpha+\beta}{2}-1} K_{\alpha-\beta}(2\sqrt{\alpha\beta}h) dh \quad (1.109)$$

According to the properties of the Meijer-G function, Eq. (1.109) can be written as a closed expression, as follows:

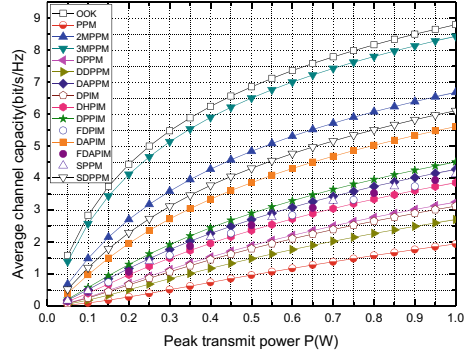
$$\langle C \rangle = \frac{B \cdot 2^{\alpha+\beta-1}}{2 \ln 2 \cdot \pi \Gamma(\alpha)\Gamma(\beta)} G_{6,2}^{1,6} \left(\frac{8t^2 P_c^2 \eta_z^2 R}{\sigma_n^2 (\alpha\beta)^2} \cdot \left(\frac{\eta A}{\lambda L} \right)^4 \middle| \begin{matrix} 1, 1, \frac{1-\alpha}{2}, \frac{2-\alpha}{2}, \frac{1-\beta}{2}, \frac{2-\beta}{2} \\ 1, 0, \frac{2-\alpha-\beta}{4}, \frac{4-\alpha-\beta}{4} \end{matrix} \right). \quad (1.110)$$

A numerical simulation of Eq. (1.110) was carried out, and the values of each parameter were consistent with those of weak turbulence. The comparison of the average channel capacity of different modulation methods in the case of moderate to strong turbulence is shown in Fig. 1.17.

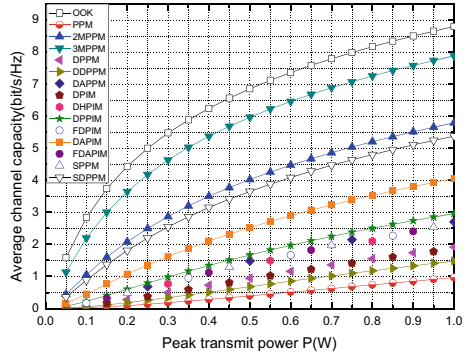
Figure 1.17 shows a comparison of the average channel capacities of various modulation schemes with different transmission distances and different modulation-bit numbers in the case of moderate to strong turbulence. From these three subfigures, it can be seen that the variation trend of the average channel-capacity under the same conditions is consistent with that of weak turbulence. When the transmission distance is 5000 m, the average channel capacity ratio of each modulation method (except OOK), when the number of modulation bits is five, is six.

When the number of modulation bits is six, the average channel capacity of each modulation method, when the transmission distance is 3000 m, is higher than that when the transmission distance is 5000 m (3–4 bits/s/Hz). The difference is that, compared with the case of weak turbulence, the average channel capacity of each

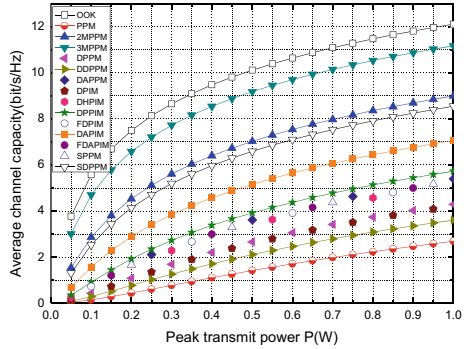
Fig. 1.17 Average channel capacity of each modulation method under moderate to strong turbulence [20]



(a) $L = 5000m, M = 5bits, C_n^2 = 9 \times 10^{-15} m^{-2/3}$



(b) $L = 5000m, M = 6bits, C_n^2 = 9 \times 10^{-15} m^{-2/3}$



(c) $L = 3000m, M = 5bits, C_n^2 = 9 \times 10^{-15} m^{-2/3}$

modulation scheme is reduced by about 1–2 bits/s/Hz under the same conditions as medium and strong turbulence.

In the case of strong turbulence, the channel state obeys a K distribution. Combining Eqs. (1.106), (1.107), and (1.102), the expression of the average channel capacity of the different modulation methods is

$$\langle C \rangle = \frac{2Ba^{(a+1)/2}}{\ln 2 \cdot \Gamma(a)} \int_0^\infty \ln \left(1 + \frac{\eta_z^2 t^2 P_c^2 R}{2\sigma_n^2} \left(\frac{\eta A}{\lambda L} \right)^4 h^2 \right) \cdot h^{\frac{a-1}{2}} K_{a-1}(2\sqrt{ah}) dh. \quad (1.111)$$

According to Eqs. (1.98), (1.99), (1.100), and (1.101), the closed expression of the average channel capacity of each modulation mode in a strongly turbulent channel can be obtained as

$$\langle C \rangle = \frac{B \cdot 2^a}{2\pi \ln 2 \Gamma(a)} G_{6,2}^{1,6} \left(\frac{8t^2 P_c^2 \eta_z^2 R}{a^2 \sigma_n^2} \cdot \left(\frac{\eta A}{\lambda L} \right)^4 \middle| \begin{matrix} 1, 1, \frac{1-a}{2}, \frac{2-a}{2}, 0, \frac{1}{2} \\ 1, 0, \frac{1-a}{4}, \frac{3-a}{4}, 0 \end{matrix} \right). \quad (1.112)$$

The numerical simulation of the above formula shows a comparison of the average channel capacity of each modulation method in the case of strong turbulence, as shown in Fig. 1.18.

Figure 1.18 shows a comparison of the average channel capacity of each modulation method under the condition of strong turbulence with different transmission distances, modulation-bit numbers, and a values. From the four sub-graphs, (a), (b), (c), and (d), it can be seen that the influence of the transmission distance and number of modulation bits on the average channel capacity of each modulation method is the same as that of weak turbulence and medium-strong turbulence. The larger the value of parameter a , the larger the average channel capacity; however, the difference is not very obvious. Compared with the case of moderate and strong turbulence, under the same conditions, the average channel capacity in the case of strong turbulence is approximately 1 bit/s/Hz smaller [20].

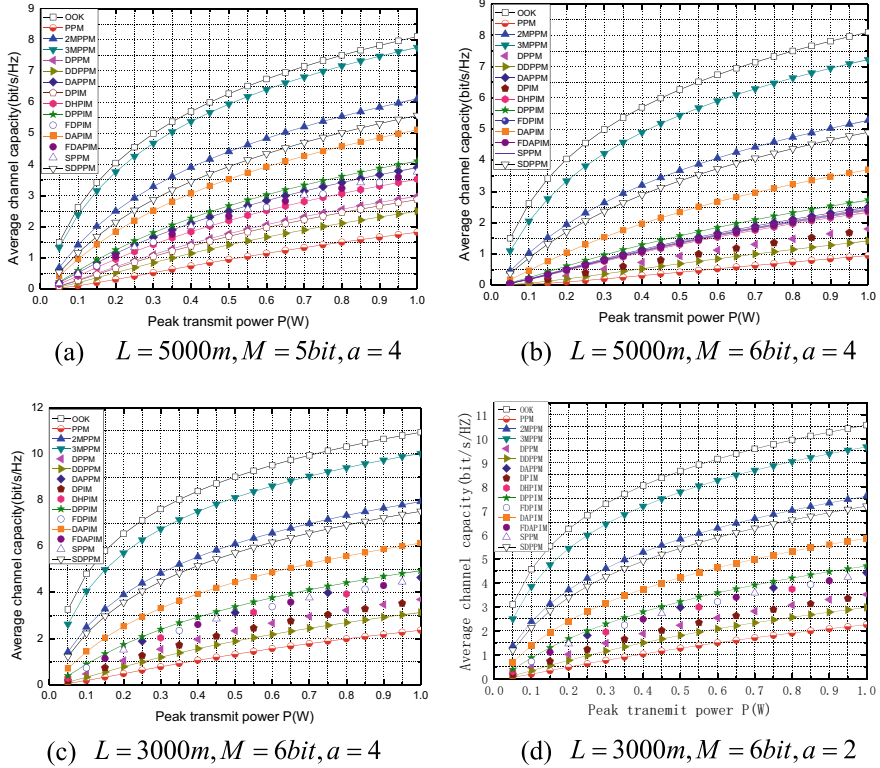


Fig. 1.18 Average channel capacity of each modulation method under strong turbulence [20]

References

1. Barry JR (1997) Wireless infrared communication. Springer, New York
2. Pierce J (1978) Optical channels: practical limits with photon counting. IEEE Trans Commun 26(12):1819–1821
3. Pease R (1999) Optical laser communication systems Carve Niche in Metro Markets. Light Wave 01(9):23–27
4. Sethakaset U, Gulliver TA (2004) Differential Amplitude Pulse-Position Modulation for Indoor Wireless Optical Channels. In: IEEE global telecommunications conference, Texas, US, 2004. pp 1867–1871
5. Sui M, Yu XS, Zhou ZG (2009) The Modified PPM Modulation for Underwater optical wireless communication. In: IEEE international conference on communication software and networks, 2009, Chengdu, China, pp 173–177
6. Guan DQ, Wei W, Du J et al (2007) Analysis of optical wireless communication technology. Ship Electron Eng 27(5):37–41
7. Aldibbiat NM, Ghassemlooy Z, McLaughlin R (2002) Dual header pulse interval modulation for dispersive in door optical wireless communication systems. IEEE Proc Circuits Devices Syst 149(3):187–192
8. Park H, Barry JR (1995) Modulation analysis for wireless infrared communication. In: IEEE, international conference on communications (ICC). Seattle, USA, pp 1182–1186

9. Zhang TY, Wang HX, Zhu YB et al (2006) Analyses for wireless optical system modulation caused by Threshold. *J Proj, Rocket, Missiles Guid* 26(4):323–325
10. Wang HX, Zhu YB, Zhang TY et al (2006) Performance study of modulation for optical wireless communication. *Laser Optoelectron Prog* 43(06):38–41
11. Wang HX, Xu JW, Sun XM (2012) Performance analysis of MIMO-FSO systems based on PPM modulation and Gamma-Gamma distribution model. *J Syst Eng Electron* 34(2):385–390
12. Majumdar AK (2005) Free-space laser communication performance in the atmospheric channel. *Optical and Fiber Communications Reports* 2(4):345–396
13. Sandalidis HG, Tsiftsis TA (2008) Outage probability and ergodic capacity of free-space optical links over strong turbulence. *Electron Lett* 44(1):46–47
14. Li J, Uysal M (2003) Optical wireless communications: system model, capacity and coding. In: 2003 IEEE 58th vehicular technology conference. Orlando, USA. pp 168–172
15. Zhu XM, Kahn JM (2002) Free-space optical communication through atmospheric turbulence channels. *IEEE Trans Commun* 50(8):1293–1300
16. Nistazakis HE, Karagianni EA, Tsigopoulos AD (2009) Average capacity of optical wireless communication system over atmospheric turbulence channels. *J Lightwave Technol* 27(8):974–979
17. Nistazakis HE, Marinos D, Haniyas M, et al (2010) Estimation of capacity bounds of free space optical channels under strong turbulence conditions. In: 18-th international conference on microwaves, radar and wireless communications. IEEE, Vilnius, Lithuania. pp 1–3
18. Wolfram. The wolfram functions site [OL]. <https://library.wolfram.com/infocenter/Conferences/5821/html> [2010-10-20]
19. Yi X, Liu Z, Yue P, et al (2010) BER performance analysis for Mary PPM over gamma-gamma atmospheric turbulence channels. In: 2010 6th international conference on wireless communications networking and mobile computing. IEEE, Chengdu, China. pp 1–4
20. Liu MP (2013) Performances analysis of PPMS' in optical wireless communication. Xi'an: Xi'an University of Technology

Chapter 2

Error Control Based on RS Codes



The RS code was first constructed by Reed and Solomon in 1960. It is one of the most effective and widely used error-control codes. It is based on the Bose–Chaudhuri–Hocquenghem (BCH) code [1–9]. This chapter starts with finite-field arithmetic, introduces the application of RS codes to wireless laser-communication systems, including RS coding and decoding, and finally discusses the hardware implementation of RS codes.

2.1 Basic Principles of Error-Control Coding

Because there is always noise in a channel, errors may occur as a result of noise affecting useful information [10]. Errors can be divided into two categories: burst and random. Burst errors are caused by burst noise. This type of error occurs in clumps; that is, if errors occur, they will occur in adjacent symbols. The other type of error is a random error; that is, an error in a certain symbol is independent and unrelated to the preceding and following symbols.

2.1.1 Basic Error-Control Methods

There are two basic types of error-control methods [10, 11]. One is called “feedback error correction” and the other is called “forward error correction.” On this basis, a method called “hybrid error correction” is derived.

(1) Feedback error correction

In this method, the sender encodes the transmitted information using a simple encoding method that can detect a certain degree of transmission errors. It adds

a small number of supervision symbols, and checks the received encoded signal, according to the encoding rules, at the receiver. If an error code is detected, an inquiry signal is sent to the sender to request a retransmission. When the sender receives the inquiry signal, it immediately retransmits the part of the sent information that had the transmission errors until it is received correctly. “Error detection” means that one or more of the received symbols are known to be wrong; however, the exact position of the error is not necessarily known.

(2) Forward error correction

In this method, the sender adopts a complex coding method that can correct a certain degree of transmission errors during decoding; thus, the receiver can not only find the error code in the received signal code, but also correct it. When forward error correction is used, no feedback channel is required, and no repeated retransmissions are needed, which would delay the transmission time. This is beneficial for real-time transmission; however, the error-correction system is more complicated.

(3) Hybrid error correction

In hybrid error correction, a small number of more serious errors are automatically corrected at the receiving end. Therefore, “hybrid error correction” is a mixture of “forward error correction” and “feedback error correction.”

Different error-control techniques should be used for different types of channels; otherwise, part of the effort will be wasted. Feedback error correction can be used for two-way data communication. Forward error correction is used for unidirectional digital signal transmissions, such as broadcast digital television systems, because such systems do not have a feedback channel.

2.1.2 Error-Control Fundamentals [10, 11]

Following is an example from everyday life. Suppose someone sends a notification: “There will be a meeting tomorrow from 14:00–16:00;” however, an error occurs during the notification process, and it becomes “There will be a meeting from 10:00–16:00 tomorrow.” When others receive this erroneous notification, they will come at the wrong time because they will assume that the message is correct.

To enable the receiver to judge whether the message is correct, the word “afternoon” can be added; that is, “The meeting will be held from 14:00–16:00 tomorrow afternoon.” If the message is still wrong: “Meeting tomorrow afternoon from 10:00–16:00,” the receiver can judge that an error occurred at “10:00” because of the word “afternoon.” However, the receiver still cannot correct the error, because it cannot determine the correct time.

At this point, the receiver can tell the sender to send the notification again, which is error detection and retransmission. To enable both error detection and error correction, “two hours” can be added to the content of the notification; that is, it is changed

to “Meet tomorrow afternoon from 14:00–16:00 for a two-hour meeting.” In this manner, if “14:00” is mistaken for “10:00,” the receiver can not only judge the error, but also correct the error, because the added “two hours” reveal the correct time to be 14:00–16:00.

The above example illustrates that to determine whether transmitted information is wrong, additional judgment data can be added during a transmission. If the error cannot be corrected, additional judgment data need to be added. These additional data are completely redundant if there are no bit errors; however, if a bit error occurs, the specific relationship between the transmitted information data and the additional data can be used to detect and correct errors.

Source code controls the basic principles of coding. Specifically, to enable the source code to have error-detection and -correction capabilities, some redundant symbols (also known as supervisory codes) should be added to the source code, according to certain rules, and a certain relationship should be established between the transmitted information symbols. When the sender completes this task, it is called error-control coding; the receiver can detect or correct errors, according to the specific relationship between the information symbols and the supervisory symbols. If something is wrong, the original information symbol is output, and the process of completing this task is called error-control decoding (or decoding).

In addition, regardless of error detection and correction, there is a certain range for misjudgment. In the above example, if the meeting time is incorrectly set to “16:00–18:00,” the error cannot be detected and corrected, because this time also satisfies the constraints of the additional data, which means more data should be added (i.e., redundancy).

We know that a central task of programming is to eliminate redundancy and reduce the code rate. However, to detect and correct errors, redundancy must be added, which will inevitably lead to an increase in the code rate and a decrease in transmission efficiency. Obviously, this is a contradiction.

Our purpose is to find a better encoding method that can enable error detection and correction without adding too much redundancy. Furthermore, after programming, if the transmission-channel capacity matches the source rate of the signal, and the noise introduced in the channel is small, the bit-error rate is generally very low. For example, when the signal-to-noise ratio of a channel exceeds 20 dB, the bit-error rate of binary unipolar code is lower than 10^{-8} ; hence, it is possible to enable error detection and error correction through channel coding.

2.1.3 Error-Control Code Classification

With the development of digital communication technology, people have researched and developed various error-control coding schemes, each of which is based on a corresponding mathematical model and has different error-detection and -correction characteristics that can control the error code from different perspectives. According to their different functions, error control codes can be divided into error detection,

error correction, and erasure correction. Error-detection codes simply identify error codes without correcting them; error-correction codes not only identify error codes, but also correct them; erasure correction codes not only identify and correct error codes, but also delete an error that exceeds the correction range and cannot be corrected.

Error codes **can be divided into codes for correcting random errors and codes for correcting burst errors**, depending on the different causes of the errors. **The former is mainly used to address independent partial errors, while the latter is mainly used to address large-area continuous errors, such as information loss caused by magnetic powder falling off a digital tape recording.**

The relationships between an information symbol and an additional supervisory symbol can be divided into linear and nonlinear. If the relationship between the two satisfies a set of linear equations, it is called a linear code; otherwise, if their relationship cannot be described by a linear equation, it is called a nonlinear code.

Codes can be divided into block codes and convolutional codes, according to the different constraints between information symbols and additional supervisory symbols. In a block code, the encoded symbol sequence is divided into n -bit groups, including k -bit information symbols and r -bit additional supervisory symbols; that is, $n = k + r$. The information symbols of a group are related, and unrelated to the information symbols of other groups.

Convolutional codes are different. Although an encoded symbol sequence is also divided into code groups, the supervisory symbols of each group are not only related to the information symbols of the group, but also have a constraint relationship with the information symbols of the previous code group.

Codes can be divided into systematic and non-systematic codes, depending on whether the original form of the information symbol remains unchanged after encoding. In a systematic code, the encoded information-symbol sequence remains unchanged, while in a non-systematic code, the information symbol changes the original signal sequence. The decoding circuit is more complicated because of the change of the original code bits, so non-systematic code is selected less frequently.

2.2 Basic RS-Code Principles

The RS code was first constructed by Reed and Solomon in 1960. It is one of the most effective and widely used error-control codes. It is a type of multi-input code with a strong error-correction ability. It can correct burst errors and random errors, but is especially suitable for correcting burst errors. Random errors are errors caused by single transmitted bits. A burst error is a type of error that occurs centrally. The RS code is a type of non-binary BCH code. In (n, k) RS code, the input signal is divided into k m -bit groups, where each group includes k symbols, and each symbol consists of m bits. The binary code described above consists of one bit.

For an RS code with a length of $2^m - 1$ symbols, each symbol can be regarded as an element in a finite Galois field $GF(2^m)$. The generator polynomial of an RS code with a minimum code distance of d symbols has the following form:

$$g(D) = (D + \alpha)(D + \alpha^2) \cdots (D + \alpha^{d-1}). \quad (2.1)$$

Here, α^i is an element in $GF(2^m)$. For example, if an RS code was constructed that could correct three error symbols, the code length would be 15, and m would equal four. From the RS-code parameters, it can be known that the code distance of the code is seven, and the supervisory section is six symbols. Hence, the code is a (15, 9) RS code. The resulting polynomial is

$$\begin{aligned} g(D) &= (D + \alpha)(D + \alpha^2)(D + \alpha^3)(D + \alpha^4)(D + \alpha^5)(D + \alpha^6) \\ &= D^6 + \alpha^{10}D^5 + \alpha^{14}D^4 + \alpha^4D^3 + \alpha^6D^2 + \alpha^9D + \alpha^6. \end{aligned} \quad (2.2)$$

Thus, from a binary perspective, this is a (60, 36) code.

2.2.1 Basic Concept of an RS Code [12, 13]

RS codes are used to correct burst errors and are also suitable for correcting random errors. An RS code that corrects t errors has the following parameters:

- The number of bits per symbol is m .
- Each symbol can be seen as an element in the finite field $GF(2^m)$.
- The code length is $n = 2^m - 1$ symbols.
- An information segment is k symbols.
- The supervisory segment is $n - k = 2t$ symbols.
- The minimum code distance is $d = 2t + 1$ symbols.
- The burst-error pattern used by the RS code for correction is as follows:
 - A single burst has a length of $b_1 = (t - 1)m + 1$ bits.
 - Two bursts have a total length of $b_2 = (t - 3)m + 1$ bits.
 - i bursts have a length of $b_i = (t - 2i + 1)m + 2i - 1$ bits.

2.2.2 Properties of RS Codes

In non-binary codes, if p is a prime number and q is a multiple power of p , there exists a q -ary primitive code generated by the Galois field $GF(q)$. The generator polynomial of a q -ary (n, k) cyclic code is a factor of

$x^n - 1$, which is a polynomial of degree $n - k$ with coefficients over the elements in the $GF(q)$ field. For arbitrarily selected positive integers s and t , there exists a q -ary BCH code of length $n = q^s - 1$ [1]. It can correct t errors with only $2(st)$ check bits.

A q -ary BCH code is called an RS code when $s = 1$. An important property of RS codes is that the true minimum distance minus the design distance is always the same. Each RS code has a maximum distance separable code because its minimum distance is equal to $n - k + 1$. When some information symbols of the RS code are omitted, the packet length is shortened; however, the minimum distance is not reduced, so any shortened code is still a maximum-distance separable (MDS) code [1].

Another important property of an RS code is that the set of available information is at any k positions within its codeword. In an (n, k) RS code, the input information is divided into km bit groups; each group includes k symbols, and each symbol is composed of m bits. The encoding and decoding of an RS code is based on a set of symbols, rather than a single 0 or 1, which is also the reason why the RS code has a particularly strong error-correction capability. This feature makes RS codes particularly suitable for dealing with burst errors.

2.3 RS-Code Encoding Method

2.3.1 *Galois-Field Arithmetic* [1]

(1) Fundamentals

The RS code encoding and decoding algorithm is based on the mathematical concept of a Galois field. Following are some basic mathematical concepts of the Galois field and RS codes.

1. The RS code is a linear block code. In a code block with a code length of n , the first k code symbols are information bits, and the last $n-k$ code symbols are check bits;
2. The RS code is a cyclic code; that is, a codeword is still a codeword after being cyclically shifted;
3. The minimum distance $d = n - k + 1$; hence, the RS code is the maximum distance separable (MDS) code. That is, after being given n and k , the (n, k) code has the strongest error-correction ability;
4. The weight distribution of the code is completely deterministic; that is, the number of codewords with weight i in the codeword space is predictable;
5. A field consisting of a set of finite elements is called a finite field or Galois field, and the number of elements in the field is called its order;
6. A binary field consisting of two elements, 0 and 1, is called a binary finite field and denoted as $GF(2)$;
7. The smallest integer n satisfied by a non-zero element α in a finite field $GF(q)$ of order q is called the order of α .

$$\langle \alpha \rangle = \{1, \alpha, \dots, \alpha^{n-1}\}$$

An element α of order q generates a cyclic group of order q ; that is,

$$x^{q-1} - 1 = 0.$$

Any nonzero element in a finite field $\text{GF}(q)$ of order q satisfies the equation

$$x^q - x = 0. \quad (2.3)$$

8. If there is a $q-1$ -order element α in a q -order finite field $\text{GF}(q)$, α is called a primitive field element of $\text{GF}(q)$; or simply a primitive element;
9. If there is a primitive element α in the q -order finite field $\text{GF}(q)$, then all non-zero elements of $\text{GF}(q)$ constitute a $q-1$ -order cyclic group generated by α . All elements of $\text{GF}(q)$ can be explicitly expressed as

$$\text{GF}(q) = \{0, 1, \alpha, \dots, \alpha^{q-2}\}. \quad (2.4)$$

10. The q -order finite field $\text{GF}(q)$ must contain primitive elements.

(2) Polynomial and field-element arithmetic circuits [4, 6]

It can be known from matrix theory that a group of messages in a q -ary system can be expressed in the form of a polynomial; for example, 1 0 0 1 1 0 1 can be expressed as $x^6 + x^3 + x^2 + 1$.

A. Polynomial addition and subtraction circuits

Two polynomials on $\text{GF}(q)$ are known:

$$A(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1x + a_0$$

$$B(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \dots + b_1x + b_0.$$

The result of adding the above two polynomials is set to $C(x)$; then, $C(x)$ can be expressed as:

$$C(x) = A(x) + B(x) = (a_{n-1} + b_{n-1})x^{n-1} + \dots + (a_1 + b_1)x + (a_0 + b_0).$$

The above polynomial addition can be completed using the circuit shown in Fig. 2.1.

The result of adding the two polynomials, $C(x)$, is finally shifted into the $A(x)$ coefficient register. If the operation is $A(x) - B(x)$, simply replace the $B(x)$ coefficient with the addition-inverse element in $\text{GF}(q)$ to obtain $-B(x)$. Then, add $-B(x) + A(x)$.

B. Polynomial division circuit

The polynomial $B(x) = x^3 + x + 1$ divided by $A(x) = x^4 + x^3 + 1$ can be realized using the following digital circuit (Fig. 2.2).

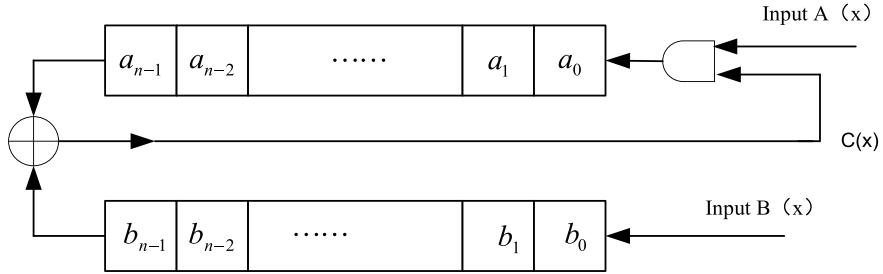


Fig. 2.1 Serial digital circuit of a polynomial adder

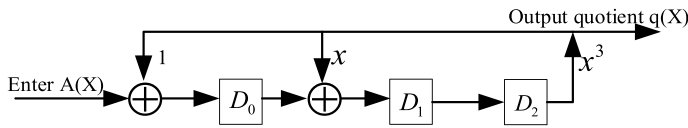


Fig. 2.2 Digital circuit of a polynomial flip-flop

2.3.2 RS-Code Encoding Methods [14–26]

RS-code coding methods can be divided into frequency-domain and time-domain coding, with corresponding frequency-domain and time-domain algorithms for decoding.

(1) Frequency-domain coding [27, 28]

[Definition] Let α be an element of order n on $GF(q)$ (n is a factor of $q-1$), and let $a(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \dots + a_1x + a_0$ be a polynomial with a degree less than n on $GF(q)$. $b(x)$ is a polynomial with $a(\alpha^j)$ ($j = 1, 2, 3, \dots, n-1$) as the coefficient. Then, $b(x) = a(\alpha^{n-1})x^{n-1} + \dots + a(\alpha) + a(1)$ is a Mattson–Solomon (MS) polynomial.

MS polynomials have the following properties:

[Property 1] If $b(x)$ is a polynomial of $a(x)$, then $a_j = n^{-1}b(\alpha^{-j})$;

[Property 2] In $GF(2^m)$, if $b(x)$ is the MS polynomial of $a(x)$, then $a_j = b(\alpha^{-j})$.

Let α be a primitive element of $GF(2^m)$, $M(x)$ be the information polynomial, and $C(x)$ be the MS polynomial of $M(x)$; then,

$$\begin{aligned}
 M(x) &= M_{k-1}x^{k-1} + M_{k-2}x^{k-2} + \dots + M_1x + M_0 \\
 C(x) &= M(\alpha^{n-1})x^{n-1} + \dots + M(\alpha)x + M(1) \\
 &= C_{n-1}x^{n-1} + C_{n-2}x^{n-2} + \dots + C_1x + C_0.
 \end{aligned} \tag{2.5}$$

In the equation $C_j = M(\alpha^j)$ ($j = 0, 1, \dots, n-1$), the vector $C = \{C_j, j = 0, 1, \dots, n-1\}$ is called the Fourier transform of the vector $M = \{M_j, j = 0, 1, \dots, n-1\}$.

According to MS polynomial property 2, $M_j = C(\alpha^{-j}) = C(\alpha^{n-j})$ ($j = 0, 1, \dots, n-1$) is due to $M_k = M_{k+1} = \dots M_{n-1} = 0$; hence, $C(\alpha^{n-k}) = C(\alpha^{n-k-1}) = \dots = C(\alpha) = 0$. This shows that $\alpha, \alpha^2, \dots, \alpha^{n-k}$ are the roots of $C(x)$; hence, $C(x)$ contains the generator polynomial $g(x)$ for RS. Thus, it is proved that the MS polynomial of the information polynomial is the code polynomial of the RS code.

(2) RS-code time-domain coding methods

Let α be a primitive element of $GF(2^m)$; then, the generator polynomial of a primitive RS code of length C to correct t errors is

$$g(x) = (x + \alpha)(x + \alpha^2) \cdots (x + \alpha^{2t}) = g_0 + g_1x + g_2x^2 + \cdots g_{2t-1}x^{2t-1} + x^{2t}. \quad (2.6)$$

Let $m(x) = m_0 + m_1x + m_2x^2 + \cdots + m_{k-1}x^{k-1}$ be the information polynomial to be encoded; then, the encoding method of the RS code is as follows:

Divide the information polynomial to be encoded by x^{n-k} bits. Then, divide the generator polynomial $g(x)$, and place the remaining x^{n-k} bits after the information polynomial to form the RS code [7]. It can be expressed as

$$c(x) = x^{2t}m(x) + x^{2t}m(x) \bmod g(x). \quad (2.7)$$

2.4 RS-Code Decoding Methods [27–29]

Assume that $c(x)$ is the transmitted signal, $r(x)$ is the received signal, and $e(x)$ is the error pattern; then,

$$r(x) = c(x) + e(x). \quad (2.8)$$

It can be seen from Eq. (2.8) that if the value of the error pattern can be obtained, the correct information $c(x)$ can be obtained from the received signal $r(x)$. The general decoding algorithm steps are as follows:

1. Obtain the syndrome S_j , according to the received codeword;
2. Find the error-position polynomial from the syndrome;
3. Find the error-position value from the error-position polynomial;
4. Calculate the corresponding error value from the error-position value;
5. The error pattern is obtained, and the error correction is successful.

The RS code has only a single encoding method, but many types of decoding algorithms; they are primarily frequency and time domain. Calculating the error

polynomial is the focus and challenge of the problem. The specifics of the Berlekamp–Massey and Forney algorithms will be given in the following sections.

2.4.1 RS-Code Frequency-Domain Decoding Algorithms [27–29]

Let $r = \{r_i; i = 0, 1, 2, \dots, n-1\}$ be a vector on the $\text{GF}(2^m)$ domain, and a primitive element of $\text{GF}(2^m)$. $\text{GF}(2^m)$ is the finite-field Fourier transform of vector r . $R = \{R_j; j = 0, 1, \dots, n-1\}$ is another vector; then,

$$R_j = r(\alpha^j) \quad (j = 0, 1, 2, \dots, n-1).$$

$$r_i = R(\alpha^{-i}) \quad (i = 0, 1, 2, \dots, n-1).$$

Following are the decoding steps:

1. Calculate the Fourier transform R of the received vector r to obtain $2t$ syndromes, $S_j = R_j$;
2. Use the Berlekamp–Massey (BM) algorithm to find the error-position polynomial $\sigma(x)$;
3. Calculate E from $\sigma(x)$ and S_1, S_2, \dots, S_{2t} ;

When $j = 1, 2, \dots, 2t$, $E_j = R_j$ and the rest of E_j can be derived from the following formula:

$$E_j + E_{j-1}\sigma_1 + E_{j-2}\sigma_2 + \dots + E_{j-e}\sigma_e = 0. \quad (2.9)$$

4. The correct codeword C is obtained using the inverse transformation of $R-E$, and the information vector M is obtained by the inverse Fourier transform of C .

(1) Concomitant search

From the previous error-pattern equation, if t errors are generated, then

$$E(x) = Y_t x^{l_t} + Y_{t-1} x^{l_{t-1}} + \dots + Y_1 x^{l_1} = \sum_{i=1}^t Y_i x^{l_i}, \quad (2.10)$$

where x^{l_i} is the number of error locations, and Y_i is the error value. From the definition of the accompanying equation, it can be known that

$$\begin{aligned}
S^T &= H \bullet R^T = H \bullet E^T \\
&= \begin{bmatrix} (\alpha^{m_0})^{n-1} & (\alpha^{m_0})^{n-2} & \cdots & \alpha^{m_0} & 1 \\ (\alpha^{m_0+1})^{n-1} & (\alpha^{m_0+1})^{n-2} & \cdots & \alpha^{m_0+1} & 1 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ (\alpha^{m_0+2t-1})^{n-1} & (\alpha^{m_0+2t-1})^{n-2} & \cdots & \alpha^{m_0+2t-1} & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ Y_t \\ \vdots \\ 0 \\ 0 \\ \vdots \\ Y_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} Y_1(\alpha^{m_0})^{l_{i1}} + Y_2(\alpha^{m_0})^{l_{i2}} + \cdots + Y_t(\alpha^{m_0})^{l_{it}} \\ Y_1(\alpha^{m_0+1})^{l_{i1}} + Y_2(\alpha^{m_0+1})^{l_{i2}} + \cdots + Y_t(\alpha^{m_0+1})^{l_{it}} \\ \vdots \\ Y_1(\alpha^{m_0+2t-1})^{l_{i1}} + Y_2(\alpha^{m_0+2t-1})^{l_{i2}} + \cdots + Y_t(\alpha^{m_0+2t-1})^{l_{it}} \end{bmatrix} \\
&= \begin{bmatrix} s_{m_0} = R(\alpha^{m_0}) = E(\alpha^{m_0}) \\ s_{m_0+1} = R(\alpha^{m_0+1}) = E(\alpha^{m_0+1}) \\ \vdots \\ s_{m_0+2t-1} = R(\alpha^{m_0+2t-1}) = E(\alpha^{m_0+2t-1}) \end{bmatrix}.
\end{aligned}$$

Here, it is expressed as an equation:

$$\begin{aligned}
s_1 &= Y_1x_1 + Y_2x_2 + \cdots + Y_tx_t = \sum_{k=1}^t Y_kx_k \\
s_2 &= Y_1x_1^1 + Y_2x_2^2 + \cdots + Y_tx_t^2 = \sum_{k=1}^t Y_kx_k^2 \\
&\vdots \\
s_{2t} &= Y_1x_1^{2t} + Y_2x_2^{2t} + \cdots + Y_tx_t^{2t} = \sum_{k=1}^t Y_kx_k^{2t}
\end{aligned} \tag{2.11}$$

The s_j in the formula is called the weighted power and symmetric function of x .

(2) Solving for the number of wrong positions

We introduce an error-position polynomial:

$$\sigma(x) = (1 - x_1x)(1 - x_2x) \cdots (1 - x_tx). \quad (2.12)$$

We expand it to

$$\sigma(x) = \prod_{i=1}^t (1 - x_ix) = 1 + \sigma_1x + \sigma_2x^2 + \cdots + \sigma_tx^t. \quad (2.13)$$

If x_k^{-1} is the wrong location, then

$$\sigma(x_k^{-1}) = 1 + \sigma_1x_k^{-1} + \cdots + \sigma_tx_k^{-t} = 0. \quad (2.14)$$

We multiply both sides by $Y_kx_k^{j+t}$, $j = m_0, m_0 + 1, \cdots, m_0 + t - 1$, and sum k to obtain

$$\sum_{k=1}^t Y_kx_k^{j+t} + \sigma_1 \sum_{k=1}^t Y_kx_k^{j+t-1} + \cdots + \sigma_t \sum_{k=1}^t Y_kx_k^j = 0 \quad k = 1, 2, \cdots, t. \quad (2.15)$$

From the definition of the syndrome s_j , it can be seen that the above equation can be written as

$$s_{j+t} + \sigma_1s_{j+t-1} + \cdots + \sigma_ts_j = 0. \quad (2.16)$$

Expanding the above equation into matrix form produces

$$\begin{bmatrix} s_{m_0+t-1} & s_{m_0+t-2} & \cdots & s_{m_0} \\ s_{m_0+t} & s_{m_0+t-1} & \cdots & s_{m_0+1} \\ \vdots & \vdots & \vdots & \vdots \\ s_{m_0+2t-2} & s_{m_0+2t-3} & \cdots & s_{m_0+t-1} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_t \end{bmatrix} = - \begin{bmatrix} s_{m_0+t} \\ s_{m_0+t-1} \\ \vdots \\ s_{m_0+2t-1} \end{bmatrix}.$$

It can also be expressed as

$$[M][\sigma] = -[S]. \quad (2.17)$$

If the number of errors is t , the matrix M is full rank, and we can find σ from the above equation. If $\det(m) = 0$, we reduce the matrix dimension by 1 and repeat the steps until a solution is found. Finally, the number of errors t and σ can be obtained simultaneously. Then, $\sigma(x)$ is solved according to the money-search method, described later, and the error location and error value are obtained [17].

2.4.2 RS-Code Domain-Decoding Algorithm

Suppose the transmitted codeword is $c(x) = c_0 + c_1x + \cdots + c_{n-1}x^{n-1}$, and the received codeword is

$$r(x) = r_0 + r_1x + \cdots + r_{n-1}x^{n-1}. \quad (2.18)$$

Let $e(x)$ be the error pattern: $e(x) = e_0 + e_1x + \cdots + e_{n-1}x^{n-1}$; then,

$$r(x) = c(x) + e(x). \quad (2.19)$$

There can be at most t errors in $e(x)$; otherwise, the errors are uncorrectable. Assuming that e errors ($0 \leq e \leq t$) actually occurred, and they occurred at positions i_1, i_2, \dots, i_e ; then,

$$e(x) = y_1x^{i_1} + y_2x^{i_2} + \cdots + y_ex^{i_e}, \quad (2.20)$$

where $x^{i_1}, x^{i_2}, \dots, x^{i_e}$ is the number of error positions, and y_1, y_2, \dots, y_e are the corresponding error values.

The task of decoding is to obtain the number of error positions and the corresponding error values from the received vector $r(x)$. The decoding steps follow:

1. Calculate the syndrome ($j = 1, 2, \dots, 2t$) of the received vector $r(x)$.
2. Using the Berlekamp–Massey iterative algorithm, the error-location polynomial $\sigma(x)$ is obtained from the syndrome S_j .
3. Find the reciprocal root of $\sigma(x)$ and determine the location of the error, because

$$\sigma(x) = (1 + x_1x)(1 + x_2x) \cdots (1 + x_ex).$$

Among them, x_1, x_2, \dots, x_e is the reciprocal root of $\sigma(x)$. Using the heuristic method, we substitute all elements of $\text{GF}(2^m)$ into $\sigma(x)$ in turn, and the root of $\sigma(x)$ can be obtained.

4. We use the Forney algorithm to find the magnitude of the error ($j = 1, 2, \dots, e$), and obtain $e(x)$. Then, we calculate $r(x) + e(x)$ to obtain the correct codeword.

2.5 Correlation Algorithms for RS-Code Decoding

Many decoding algorithms have been created for RS codes, and many scholars are now engaged in research in this area. We discuss four of the more mature algorithms currently under study [26].

2.5.1 Peterson–Gorenstein–Zierler Decoding Algorithm

Suppose $[n, k, d]$ RS codes on $\text{GF}(q)$ can correct up to t errors, and α is a primitive element in the finite field $\text{GF}(2^m)$. The transmitted codeword has an error during channel transmission, and the error pattern is

$$e(x) = e_0 + e_1x + \dots + e_{n-1}x^{n-1}. \quad (2.21)$$

If v errors ($0 \leq v \leq t$) actually occur, the error polynomial can be rewritten as follows:

$$e(x) = y_1x^{l_1} + y_2x^{l_2} + \dots + y_vx^{l_v}. \quad (2.22)$$

According to the definition of the syndrome and the RS-code check polynomial, we obtain

$$\begin{aligned} S^T &= H \dots R^T = H \dots E^T \\ &= \begin{bmatrix} (\alpha^{m_0})^{n-1} & (\alpha^{m_0})^{n-2} & \dots & \alpha^{m_0} & 1 \\ (\alpha^{m_0+1})^{n-1} & (\alpha^{m_0+1})^{n-2} & \dots & \alpha^{m_0+1} & 1 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ (\alpha^{m_0+2t-1})^{n-1} & (\alpha^{m_0+2t-1})^{n-2} & \dots & \alpha^{m_0+2t-1} & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ Y_t \\ \vdots \\ 0 \\ 0 \\ \vdots \\ Y_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} Y_1(\alpha^{m_0})^{l_{t1}} + Y_2(\alpha^{m_0})^{l_{t2}} + \dots + Y_t(\alpha^{m_0})^{l_{tt}} \\ Y_1(\alpha^{m_0+1})^{l_{t1}} + Y_2(\alpha^{m_0+1})^{l_{t2}} + \dots + Y_t(\alpha^{m_0+1})^{l_{tt}} \\ \vdots \\ Y_1(\alpha^{m_0+2t-1})^{l_{t1}} + Y_2(\alpha^{m_0+2t-1})^{l_{t2}} + \dots + Y_t(\alpha^{m_0+2t-1})^{l_{tt}} \end{bmatrix}. \end{aligned} \quad (2.23)$$

Expressed as an equation, it is

$$\begin{aligned}
s_1 &= Y_1x_1 + Y_2x_2 + \cdots + Y_tx_t = \sum_{k=1}^t Y_kx_k \\
s_2 &= Y_1x_1^2 + Y_2x_2^2 + \cdots + Y_tx_t^2 = \sum_{k=1}^t Y_kx_k^2 \\
&\vdots \\
s_{2t} &= Y_1x_1^{2t} + Y_2x_2^{2t} + \cdots + Y_tx_t^{2t} = \sum_{k=1}^t Y_kx_k^{2t}
\end{aligned} \tag{2.24}$$

We need to solve $2v$ unknowns from these $2t$ nonlinear equations. According to the syndrome definition, the equation system has at least one set of roots; however, owing to the nonlinearity, it is not easy to solve directly. Thus, we consider other methods to solve it.

We define the error-location polynomial:

$$\Lambda(x) = \prod_{l=1}^v (1 - xX_l) = 1 + \Lambda_1x + \Lambda_2x^2 + \cdots + \Lambda_vx^v. \tag{2.25}$$

Multiplying both sides of Eq. (2.22) by $Y_lX_l^{j+v}$, and setting $x = X_l^{-1}$, we have

$$0 = Y_lX_l^{j+v}(1 + \Lambda_1X_l^{-1} + \Lambda_2X_l^{-2} + \cdots + \Lambda_vX_l^{-v}) \tag{2.26}$$

or

$$0 = Y_l(X_l^{j+v} + \Lambda_1X_l^{j+v-1} + \Lambda_2X_l^{j+v-2} + \cdots + \Lambda_vX_l^j). \tag{2.27}$$

The above formula holds for all $1 \leq l \leq v, j \in \mathbb{Z}$. We sum from 1 to v to obtain

$$\sum_{l=1}^v Y_l(X_l^{j+v} + \Lambda_1X_l^{j+v-1} + \Lambda_2X_l^{j+v-2} + \cdots + \Lambda_vX_l^j) = 0 \tag{2.28}$$

or

$$\sum_{l=1}^v Y_lX_l^{j+v} + \Lambda_1 \sum_{l=1}^v Y_lX_l^{j+v-1} + \Lambda_2 \sum_{l=1}^v Y_lX_l^{j+v-2} + \cdots + \Lambda_v \sum_{l=1}^v Y_lX_l^j = 0. \tag{2.29}$$

Therefore, the above formula can be written as

$$S_{j+v} + \Lambda_1S_{j+v-1} + \Lambda_2S_{j+v-2} + \cdots + \Lambda_vS_j = 0, j = 1, 2, \cdots, v. \tag{2.30}$$

When $v \leq \lfloor \frac{n-k}{2} \rfloor$, syndrome $S_j (0 \leq j \leq v)$ is known; hence, we can form a linear system of equations with respect to the error-location polynomial coefficients:

$$\begin{bmatrix} S_1 & S_2 & S_3 & \cdots & S_{v-1} & S_v \\ S_2 & S_3 & S_4 & \cdots & S_v & S_{v+1} \\ S_3 & S_4 & S_5 & \cdots & S_{v+1} & S_{v+2} \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ S_v & S_{v+2} & S_{v+3} & \cdots & S_{2v-2} & S_{2v-1} \end{bmatrix} \begin{bmatrix} \Lambda_v \\ \Lambda_{v-1} \\ \Lambda_{v-2} \\ \vdots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} -S_{v+1} \\ -S_{v+2} \\ -S_{v+3} \\ \vdots \\ -S_{2v} \end{bmatrix}. \quad (2.31)$$

This system of equations has v equations and v unknowns, and the necessary and sufficient condition for having a solution is that its coefficient matrix is full rank.

Let the notation $M(v)$ represent the coefficient matrix. If the codeword can correct at most t errors, it can be proved that, when the actual number of errors $v < t$, $\det(M(v)) \neq 0$ of the coefficient matrix; when $v = t$, $\det(M(v)) = 0$ and the matrix is invertible. This is the basis of the Peterson–Gorenstein–Zierler decoding algorithm.

First, let $v = t$; if $\det(M(v)) \neq 0$, then v is the number of errors; otherwise, reduce the value of v by one, and then find $\det(M(v))$. When the matrix is reduced to the $v \times v$ full rank position, find the number of errors; then, find the inverse of $M(v)$, and calculate $\Lambda(x)$. Then, use the money search to find the error position and calculate the error value. The decoding-algorithm flowchart is shown in Fig. 2.3.

It can be seen from the above decoding algorithm that the calculation amount of Eq. (2.31) is relatively large. Its calculation amount is proportional to the cube of the order of the coefficient matrix. When the code length is long and the error-correction capability is large, the amount of calculation is large. In addition, when the actual error $v < t$, the amount of calculation increases. In many practical applications, codewords that can correct many errors are often required. Therefore, this inversion algorithm is not very practical in engineering. To avoid solving the inverse of the matrix, an iterative decoding algorithm is proposed [12].

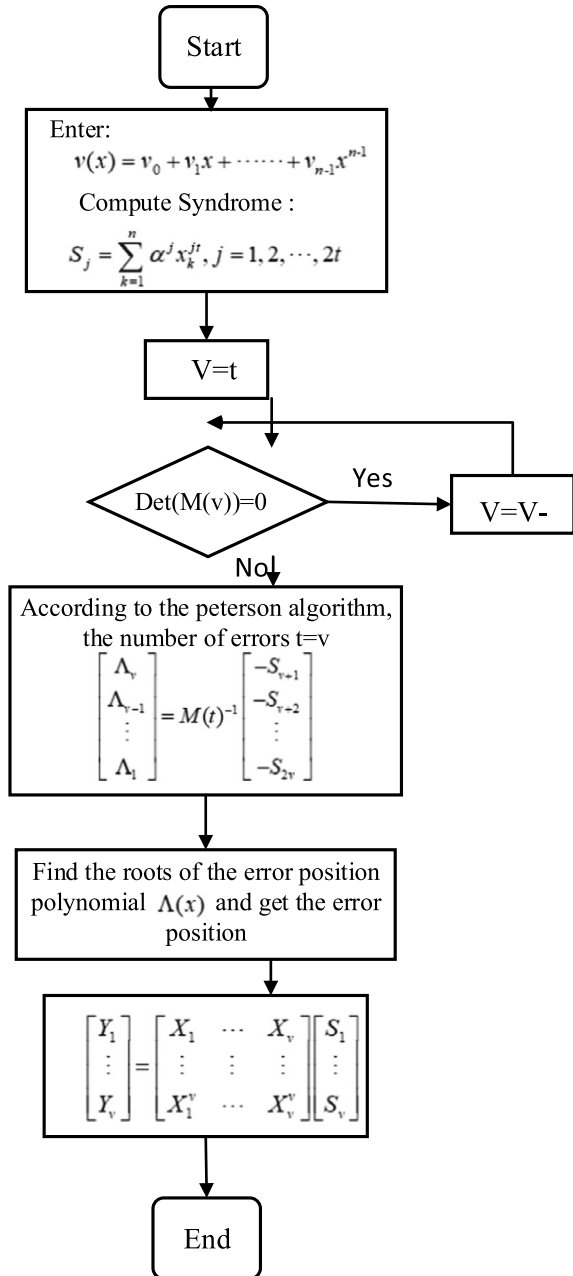
2.5.2 Berlekamp–Massey (BM) Decoding Algorithm [23]

The best way to understand this algorithm is to think of it as a problem of designing a sequence of linear-feedback shift registers. Assuming that we know the coefficients of the error-location polynomial, it can be seen from the matrix that the first row uses S_1, S_2, \dots, S_v terms for S_{v+1} , and the first row represents S_{v+1} . Two rows are represented by S_2, S_3, \dots, S_{v+1} for S_{v+2} , etc. The sequence can be represented by the following equation:

$$S_j = - \sum_{i=1}^v \Lambda_i S_{j-i} \quad j = v+1, v+2, \dots, 2v. \quad (2.32)$$

For a fixed A , the equation is an autoregressive filter whose structure is shown in Fig. 2.4.

Fig. 2.3
Peterson–Gorenstein–Zierler
decoding algorithm



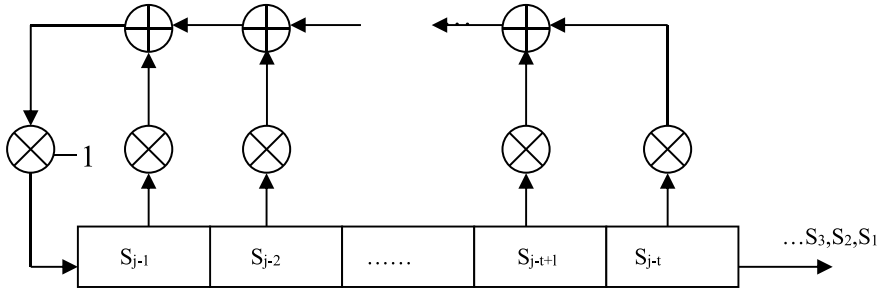


Fig. 2.4 Digital circuit diagram of an autoregressive filter

The problem then turns into designing a linear shift-register sequence that will produce these known syndrome sequences. However, many such shift-register sequences exist. We are looking for the shortest one to determine the error pattern with the least weight. That is, if $\Lambda(x)$ has the smallest degree, and the degree of the polynomial is v , there is only one polynomial of degree v .

In summary, the process of solving the error polynomial $\Lambda(x)$ is the process of constructing a minimum-length linear shift register. A linear shift register can be uniquely determined by the length L and the feedback coefficient polynomial $\Lambda(x)$. Therefore, we can denote a linear shift register as $(L, \Lambda(x))$. The idea of the BM iterative-decoding algorithm is as follows:

Assuming that the syndrome sequence is S_1, S_2, \dots, S_r , we want to generate the shortest linear shift register $(L_r, \Lambda^r(x))$ that can produce the sequence, and we have constructed a series of shift registers $(L_1, \Lambda^1(x)), (L_2, \Lambda^2(x)), \dots, (L_{r-1}, \Lambda^{r-1}(x))$. Now, the question is how to calculate the shortest shift register $(L_r, \Lambda^r(x))$, based on the known sequence of registers. In the r^{th} iteration, the next output of the $(r-1)$ shift register is:

$$\hat{S}_r = - \sum_{l=1}^{n-1} \Lambda_l^{(r-1)} S_{r-l}. \quad (2.33)$$

Subtract \hat{S} from S_r to be output to obtain the deviation Δ_r ; that is, the r^{th} deviation is

$$\Delta_r = S_r - \hat{S}_r = S_r + \sum_{l=1}^{n-1} \Lambda_l^{(r-1)} S_{r-l} = \sum_{l=0}^{n-1} \Lambda_l^{(r-1)} S_{r-l}. \quad (2.34)$$

If $\Delta_r = 0$, let $(L_r, \Lambda^r(x)) = (L_{r-1}, \Lambda^{r-1}(x))$ and the r^{th} iteration ends; if $\Delta_r \neq 0$, the shift register is modified to

$$\Lambda^{(r)}(x) = \Lambda^{(r-1)}(x) + Ax^l \Lambda^{(m-l)}(x), \quad (2.35)$$

where A is the field element, l is an integer, and $\Lambda^{(m-l)}(x)$ is a shift register polynomial that was constructed earlier. Then, we calculate the deviation of $r + l \Delta_r$, and start a new iteration round. For the detailed calculation of parameters A , l , and m , please refer to [25]. This leads to the following lemma:

Lemma 10.1 (Berlekamp–Massey algorithm) [8]: Let S_1, S_2, \dots, S_{2t} be a given field element. Using the initial conditions $\Lambda^{(0)}(x) = 1$, $B^{(0)}(x) = 1$, $L_0 = 0$, and the following recurrence relation, we can find $\Lambda^{(2t)}(x)$:

$$\begin{aligned} \Delta_r &= \sum_{j=0}^{n-1} \Lambda_j^{r-1} S_{r-j} \\ L_r &= \delta_r(r - L_{r-1}) + (1 + \delta_r)L_{r-1} \quad r = 1, 2, \dots, 2t \\ \begin{bmatrix} \Lambda^{(r)}(x) \\ B^{(r)}(x) \end{bmatrix} &= \begin{bmatrix} 1 & -\Delta_r x \\ \Delta_r^{-1} \delta_r & (1 - \delta_r)x \end{bmatrix} \begin{bmatrix} \Lambda^{(r-1)}(x) \\ B^{(r-1)}(x) \end{bmatrix} \end{aligned} \quad (2.36)$$

$$\text{where } \delta_r = \begin{cases} 1, & \text{if } \Delta_r \neq 0 \text{ and } 2L_{r-1} \leq r - 1 \\ 0, & \text{else} \end{cases}.$$

Then, $\Lambda^{(2t)}(x)$ is the lowest-degree polynomial with properties $\Lambda_0^{(2t)} = 1$ and $\sum_{j=0}^{n-1} \Lambda_j^{(2t)} S_{r-j} = 0$, $r = L_{2t} + 1, \dots, 2t$. The flowchart of the Berlekamp–Massey algorithm for solving for the error polynomial is shown in Fig. 2.5.

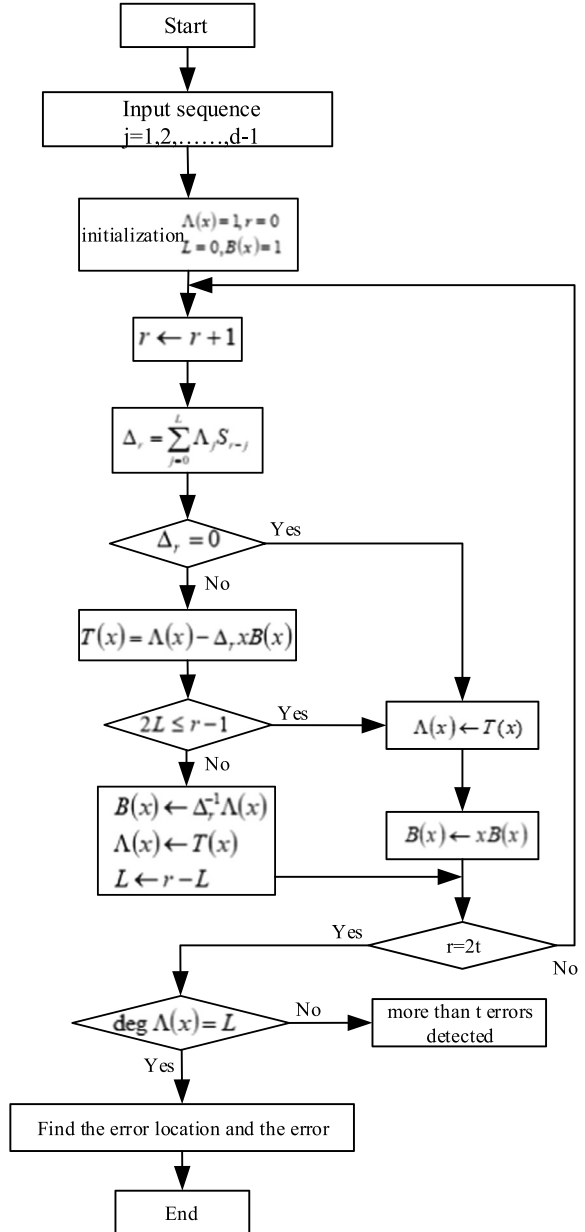
2.5.3 Qian's Search Method

After obtaining the error-position polynomial $\Lambda(x)$ with the above two algorithms, the next step is to simply find its root; that is, its error position. In 1996, Qian Wentian proposed a practical method for solving the root [25], which solved this problem.

Solving the roots of $\Lambda(x)$ is determining which bits of the receiver polynomial are in error. Let the receiver polynomial be $R(x) = r_0 + r_1x + \dots + r_{n-2}x^{n-2} + r_{n-1}x^{n-1}$. Checking whether the first bit r_{n-1} is wrong is equivalent to having the decoder determine whether α^{n-1} is the number of wrong positions, which is equivalent to checking whether $\alpha^{-(n-1)}$ is the root of $\Lambda(x)$.

That is, if $\Lambda(\alpha^{-(n-1)}) = \Lambda(\alpha) = 1 + \Lambda_1\alpha + \Lambda_2\alpha^2 + \dots + \Lambda_v\alpha^v = 0$, then, $\alpha^{-(n-1)}$ is the root of the error-position polynomial, indicating that r_{n-1} has an error. If $\Lambda(\alpha^{-(n-1)}) \neq 0$, then, $\alpha^{-(n-1)}$ is not the root of the error polynomial, and r_{n-1} is correct. Similarly, to translate r_{n-l} ($l = 1, 2, \dots, n$), the decoder must calculate $\Lambda_l\alpha^l, \Lambda_2\alpha^{2l}, \dots, \Lambda_v\alpha^{vl}$ and their sum. If $1 + \Lambda_l\alpha^l + \Lambda_2\alpha^{2l} + \dots + \Lambda_v\alpha^{vl} = 0$, then, r_{n-l} is wrong; otherwise, it is correct. In this manner, each r_{n-l} ($l = 1, 2, \dots, n$) is checked once, and the root of $\Lambda(x)$ is obtained. This process is called the money search.

Fig. 2.5
Berlekamp–Massey
algorithm flowchart



2.5.4 Forney's Algorithm

The final step in decoding is finding the error value. This step can be omitted for binary encoding, but is required for multi-binary encoding. In the Peterson–Gorenstein–Zierler algorithm, solving for the error value also requires computing the inverse of the matrix. To avoid this type of operation, Forney proposed the following algorithm.

Let the error-location polynomial be

$$\Lambda(x) = \prod_{l=1}^v (1 - xX_l) = 1 + \Lambda_1x + \Lambda_2x^2 + \cdots + \Lambda_vx^v. \quad (2.37)$$

We define the syndrome polynomial as

$$S(x) = \sum_{j=1}^{2t} S_jx^{j-1} = \sum_{j=1}^{2t} \sum_{l=1}^v Y_l X_l^j x^{j-1}. \quad (2.38)$$

We also define the error-estimation polynomials:

$$\Omega(x) = S(x)\Lambda(x) \pmod{x^{2t}}. \quad (2.39)$$

We substitute A and B into the above equation to obtain

$$\Omega(x) = \sum_{l=1}^v Y_l X_l \prod_{j \neq l} (1 - X_j x).$$

The expression for finding the error value is given below. It is much simpler than finding the inverse of a matrix.

Lemma 10.2 (Forney's algorithm) [8]: The error value can be found as follows:

$$Y_l = -\frac{X_j^{-1} \Omega(X_l^{-1})}{\prod_{j \neq l} (1 - X_j X_l^{-1})} = -\frac{\Omega(X_j^{-1})}{\Lambda'(X_l^{-1})}, l = 1, 2, \dots, t, \quad (2.40)$$

where $\Lambda'(x)$ represents the formal derivative of $\Lambda(x)$ B .

2.5.5 Euclidean Decoding Algorithm [2]

The Euclidean decoding algorithm can also be used as an RS-code decoding algorithm. Different decoders can be designed using this method and the algorithms discussed above. These decoders are easy to understand. However, their efficiency is

not high in practical applications, and efficiency is a problem that needs to be considered in practice. This decoding algorithm can simultaneously obtain polynomials $\Lambda(x)$ and $\Omega(x)$, which are expressed as follows:

Lemma 10.3 (Euclidean Algorithm) [8]: Let there be two polynomials, $s(x)$ and $t(x)$, and $\deg(s(x))\deg(t(x))$. Let $S^{(0)}(x) = s(x)$, $t^{(0)}(x) = t(x)$, $A^{(0)}(z) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, recursively, according to the following formula:

$$\begin{aligned} Q^{(r)}(x) &= \left\lfloor \frac{s^{(r)}(x)}{t^{(r)}(x)} \right\rfloor, \\ A^{(r+1)}(x) &= \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)}(x) \end{bmatrix} A^{(r)}(x), \\ \begin{bmatrix} s^{(r+1)}(x) \\ t^{(r+1)}(x) \end{bmatrix} &= \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)}(x) \end{bmatrix} \begin{bmatrix} s^{(r)}(x) \\ t^{(r)}(x) \end{bmatrix}. \end{aligned}$$

We satisfy $s^{(R)}(x) = \gamma \text{GCD}[s(x), t(x)]$ and for some values of γ , satisfy

$$\gamma \text{GCD}[s(x), t(x)] = A_{11}^{(R)}(x)s(x) + A_{12}^{(R)}(x)t(x), \quad (2.41)$$

where R is the value of r that makes $t^{(r)}(x) = 0$. In the iterative process, when $\deg t^{(R)}(x) < [(d-1)/2]$, stop the recursion. Take $\Omega(x) = t^{(R)}(x)$ and $\Lambda(x) = A_{22}^{(R)}(x)$.

The decoding-algorithm block diagram shown in Fig. 2.6 was obtained from Lemma 10.2.

In addition to the above two traditional RS decoding algorithms, new high-efficiency decoding algorithms have been developed to adapt to specific application fields. Based on Forney's generalized minimum-distance decoding (GMD) algorithm, Jamali proposed the successive-erasure minimum-distance decoding (SEMDD) algorithm. It uses the maximum-likelihood metric, which is close to the performance that the maximum-likelihood decoding method can achieve; however, the coding gain is large.

Xu proposed the maximum-likelihood deletion decoding algorithm. The basis of the algorithm is to divide the decoding generator matrix and calculate the Vandermonde system equation. The complexity of the algorithm is low. Both algorithms are suitable for short codes.

Taipale developed a computationally efficient algorithm, which can be used in RS-code soft-decision decoding processes, such as the GMD algorithm. This algorithm does not need to assume any set of $d-1$ deleted-position elements.

Vardy sets the binary BCH code equivalent to the RS code, and decodes the RS code by decoding the BCH code. The algorithm is only suitable for short RS codes at present, and its significance lies in realizing the soft-decision decoding of the RS code. Chen proposed an RS-code decoding algorithm whose burst-error correction

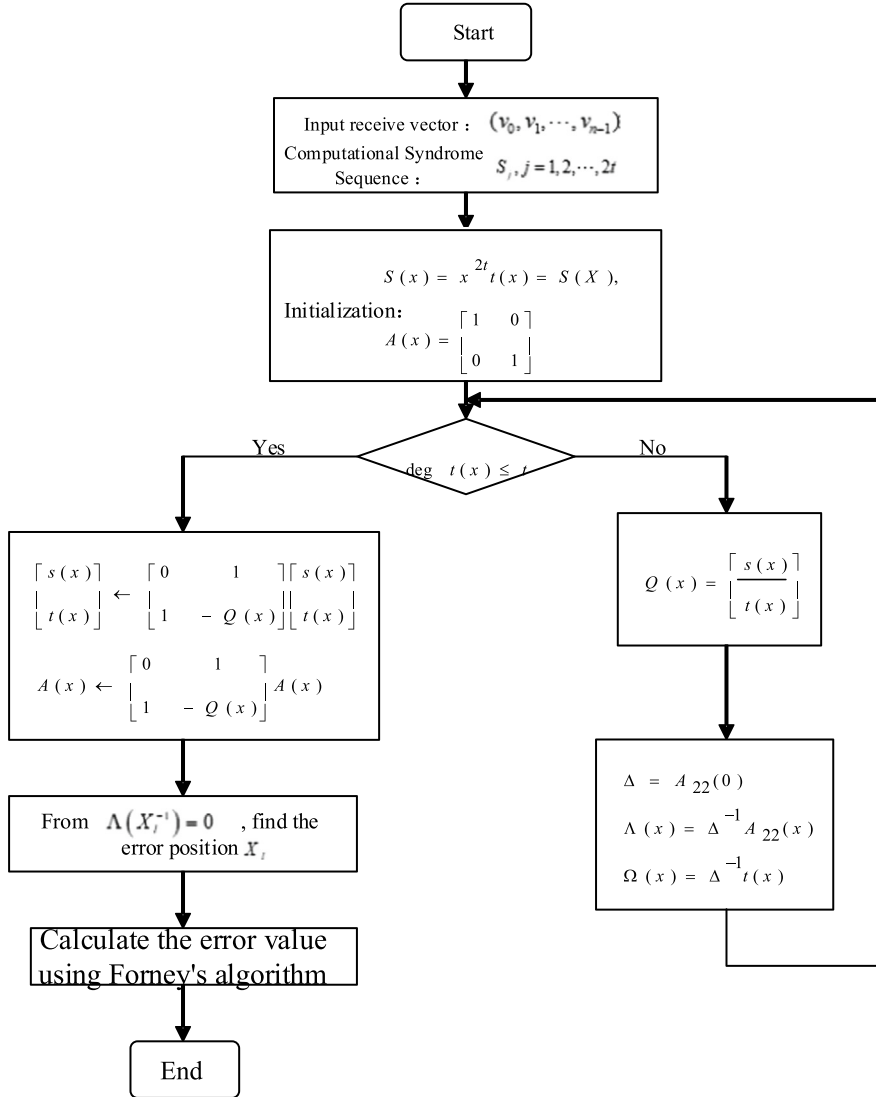


Fig. 2.6 Euclidean-algorithm flowchart

length exceeds the traditional decoding limit of $\frac{n-k}{2}$. With proper parameter selection, the decoding-error probability of the algorithm can be arbitrarily small [31].

In addition, some RS-decoding algorithms are based on the Welch–Berlekamp (WB) key equations. A fast parallel algorithm proposed by Dabiri does not need to calculate the adjoint polynomial, but is implemented with a systolic array. For RS codes whose Hamming distance does not exceed six, a decoding algorithm that solves a low-order equation of one variable is generally used. For example, Shayan’s

forward error-correction code RS (16, 12) decoding algorithm in the North American advanced train-control system, Deng's two-error correction/three-error-detection RS decoding algorithm, and Okano's use of ROM array design have been adopted into RS decoding algorithms.

For RS codes with long codeword lengths and high code rates, the BM algorithm and Euclid algorithm are mainly used for decoding, because they have good performance in decoding long codes, have a regular structure, and are easy to implement in very-large-scale integration (VLSI) [18].

2.6 Hardware Implementation of an RS(15, 9) Code-Encoding System

The hardware implementation of the RS code for a wireless laser-communication system was selected to be completed on a field-programmable gate array (FPGA). Because the development of programmable chip-development languages, such as Verilog hardware-description language (HDL) and VHDL, and the development systems supported by them are becoming increasingly perfect, we can fully implement a single FPGA chip to complete the required RS-code encoding and decoding methods.

2.6.1 Numerical Operations in a Finite Field $GF(2^4)$

RS error-correction coding is based on finite fields; generally, $GF(2^m)$ is used to represent the finite field. m is the number of bits in the symbol, and different m correspond to different primitive polynomials (a reduced polynomial of degree m $f(x)$). If the entire table containing 2^m different symbols of 0 and 1 can be given, it is called a primitive polynomial. The RS(15, 9) encoder designed in this study is based on a finite field of $GF(2^4)$ and the selected primitive polynomial is $4f(x) = 1 + x + x^4$. The process of calculating each element of $GF(2^4)$ is shown below:

Set $\beta = a$,

followed by $\beta^{2^0} = a$, $\beta^{2^1} = a^2$, $\beta^{2^2} = a^4$, $\beta^{2^3} = a^8$, $\beta^{2^4} = a^{16} = a, \dots \dots$

Then, we have $a^5 = a \cdot a^4 = a + a^2$,

and by analogy, $a^6, a^7, a^8, \dots a^{14}, a^{15} = 1$, $a^{16} = a \cdot a^{15} = a, \dots$

Therefore, we can obtain the $GF(2^4)$ element table shown in Table 2.1.

(1) Coding theory for the RS(15, 9) code encoder

The design parameters and technical indicators of the RS(15, 9) code encoder are as follows:

- Bits per symbol: $m = 4$;
- Code length: $n = 2^m - 1 = 15$;

Table 2.1 GF(2^4) element table

Elements of a finite field	Binary representation (a^3, a^2, a^1, a^0)	Hexadecimal representation
0	0000	0
1	0001	1
a	0010	2
a^2	0100	4
a^3	1000	8
$a^4 = a + 1$	0011	3
$a^5 = a^2 + a$	0110	6
$a^6 = a^3 + a^2$	1100	C
$a^7 = a^3 + a + 1$	1011	B
$a^8 = a^2 + 1$	0101	5
$a^9 = a^3 + a$	1010	A
$a^{10} = a^2 + a + 1$	0111	7
$a^{11} = a^3 + a^2 + a$	1110	E
$a^{12} = a^3 + a^2 + a + 1$	1111	F
$a^{13} = a^3 + a^2 + 1$	1101	D
$a^{14} = a^3 + 1$	1001	9
$a^{15} = 1$	0001	1

- Information-symbol length: $k = 9$;
- Supervisory-symbol length: $n - k = 6$;
- Error-correction ability: $(n - k)/2 = 3$;
- Yard distance: $d = 2t + 1 = 7$.

(2) Computation of Code-Generator polynomials

The calculation of two generator polynomials is introduced here.

First polynomial:

$$g(x) = LCM\{m_1(x) \cdot m_3(x) \cdot \dots \cdot m_{2t-1}(x)\}, \quad (2.42)$$

where $m_1(x), m_3(x), \dots, m_{2t-1}(x)$ is the smallest polynomial of finite field GF(2^m).

For example, $m_1(x)$ is calculated as follows:

Let $f(a) = 0$; we obtain: $1 + a + a^4 = 0$.

Because addition and subtraction are the same, $a^4 = 1 + a$.

For example:

Addition: $a^4 + a^5 = a + 1 + a^2 + a = a^2 + 1 = a^8$,

which is $(0011) + (0110) = 0101$.

Multiplication: $a^4 \cdot a^5 = a^9$,
 which is $(0011) \times (0110) = 1010$.
 Therefore,

$$m_1(x) = (x + a)(x + a^2)(x + a^4)(x + a^8). \quad (2.43)$$

With the help of finite-field elements, as shown in Table 2.10.1, we simplify $m_1(x)$ to obtain

$$m_1(x) = x^4 + x + 1. \quad (2.44)$$

We apply the same:

$$m_3(x) = x^4 + x^3 + x^2 + x + 1 \quad (2.45)$$

$$m_5(x) = x^2 + x + 1. \quad (2.46)$$

Therefore,

$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1. \quad (2.47)$$

Second polynomial:

$$g(x) = \prod_{i=1}^{2t} (x + a^i) = (x + a)(x + a^2)(x + a^3)(x + a^4)(x + a^5)(x + a^6). \quad (2.48)$$

When simplified,

$$g(x) = x^6 + a^{10}x^5 + a^{14}x^4 + a^4x^3 + a^6x^2 + a^9x + a^6. \quad (2.49)$$

From the above calculation process, we can see that the generator polynomials obtained by different calculation methods are different.

Because the generator polynomial obtained by the second method is simpler than that obtained by the first method when realized by digital circuits, we adopt the second generator polynomial.

(3) RS(15, 9)-Code coding principle

To transmit a group of information, it is first necessary to encode the group and attach a check digit. Therefore, in general, the RS-code encoding method can be summarized as follows: Divide the information polynomial to be encoded by x^{n-k} bits; then, divide by the generator polynomial $g(x)$, and place the obtained remainder after the information polynomial, increased by x^{n-k} bits to form an RS code, which can be expressed as

$$c(x) = x^{2t}m(x) + x^{2t}m(x) \bmod g(x), \quad (2.50)$$

where n is the total code length; the code length of the RS(15, 9) code is 15; k is the information-code length; the RS(15, 9) code information-code length is 9; $c(x)$ is the codeword polynomial; $m(x)$ is the information polynomial; and $g(x)$ is the generator polynomial. The generator polynomial of the RS(15, 9) code is

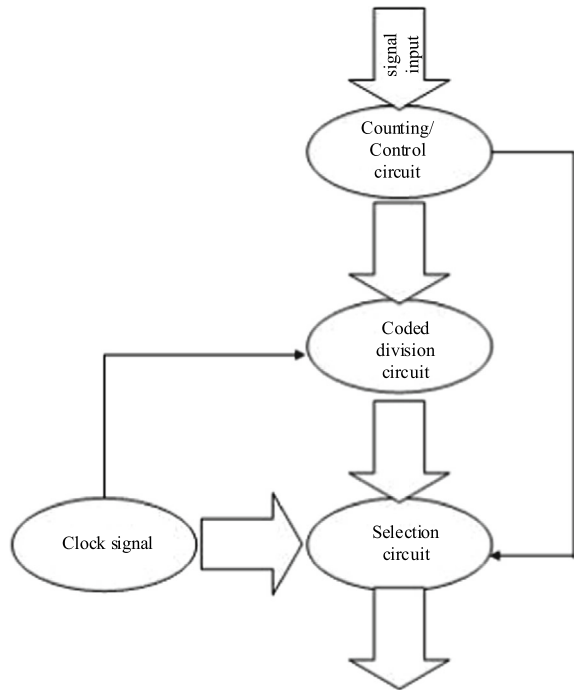
$$g(x) = x^6 + g_5x^5 + g_4x^4 + g_3x^3 + g_2x^2 + g_1x^1 + g_0. \quad (2.51)$$

2.6.2 Implementation of the Coding Circuit [6, 7, 11]

RS(15, 9) codes are a type of linear block code. We divide every nine symbols into a group, and then, according to the constraints of the generator polynomial, add six-bit supervisory symbols to the back of each group to form RS(15, 9) codes. The specific flowchart is shown in Fig. 2.7.

The operation process of the entire encoder can be seen from the flowchart. The signal first passes through the counting/control circuit and is output; then, it enters the encoding trigger circuit for operation. After all nine information bits are output, the

Fig. 2.7 RS(15, 9)-code encoding flowchart



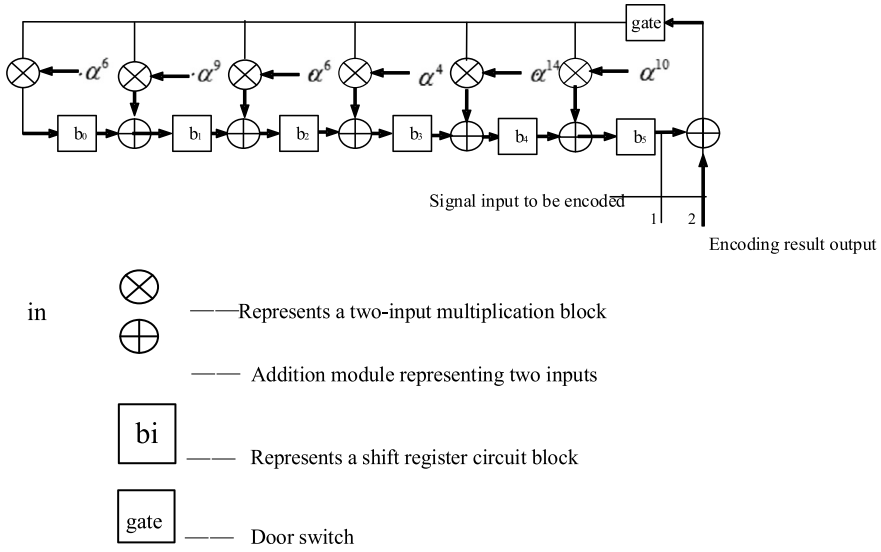


Fig. 2.8 Schematic diagram of the RS(15, 9)-code encoding circuit

six-bit result after the operation is appended. Then, the information bits are output; thus, producing a batch of 15-bit encoded codewords of nine information bits.

It can be seen from the coding polynomial $g(x)$ that the implementation of the circuit can adopt an nk -level divider structure, which includes nk -level shift registers, finite-field adders, finite-field multipliers, XOR gate structures, and a door switch. The hardware structure of the encoding circuit is shown in Fig. 2.8.

The encoding process is as follows:

1. In the initial state, the inputs are all 0, the gate is closed, and the output is connected to terminal 2. The information group is sent into the circuit in the order of (m_8, m_7, \dots, m_0) , and the output is sent into the coding-division circuit, which is equivalent to completing the multiplication of $m(x)x^{n-k}$.
2. After nine shifts, the information group is all output through the output terminal; these are the first nine information elements of the system codeword; all of them enter the division circuit to complete the remainder operation, which is stored in the shift register at this time. This produces the check digit of the codeword.
3. The gate is disconnected, and the output is connected to terminal 1. After six shifts, the parity bits $(b_5, b_4, b_3, b_2, b_1, b_0)$ in the shift register follow the output of the information bit and are sent to the receiving end. The entire codeword is $(m_8, \dots, m_0, b_5, \dots, b_0)$.
4. The gate is closed, the output is connected to terminal 2, the second group of information bits is output, and the above process is repeated.

2.6.3 Design of an RS(15, 9)-Code Decoding System

Our decoder of the RS(15, 9) code adopts the pipeline decoding algorithm. In addition, according to the characteristics of the RS(15, 9) code, the decoding circuit is simplified and optimized, and the operation rate of the system is improved [6, 7, 11, 14].

(1) Description of the basic Pipeline-Algorithm principle

The pipeline-algorithm structure is shown in Fig. 2.9, and its working principle is as follows.

After the 15-bit RS code is received, it is input into both the register and the syndrome generator. The values of the six syndromes generated by the syndrome generator are simultaneously input into the BM-algorithm loop and the inverse-transform loop. After the inverse-transform loop has calculated the first six values, the residual-transform generator, which is parallelized, sends in the next nine values. The result of the inverse-transform loop is then added (exclusive OR) to the RS code received in the register to obtain the codeword after error correction.

(2) Implementation of each part of the system

a. Design of the syndrome generator

The first step of decoding is to store the received signal r_j in the register; the value of the syndrome can be calculated using the following equation:

$$S_i = \sum_{j=0}^{15} r_j \alpha^{(k+i)j}, \quad (2.52)$$

where k is an arbitrary integer, indicating the starting period of the syndrome generator to start receiving the signal. Here, we define $k = 0$. r_j refers to the 15-codeword transmission signal that we receive every time. we start the overall calculation from the first bit, α is the element inside $GF(2^4)$.

The formula generated by the syndrome can be expressed in a more intuitive manner:

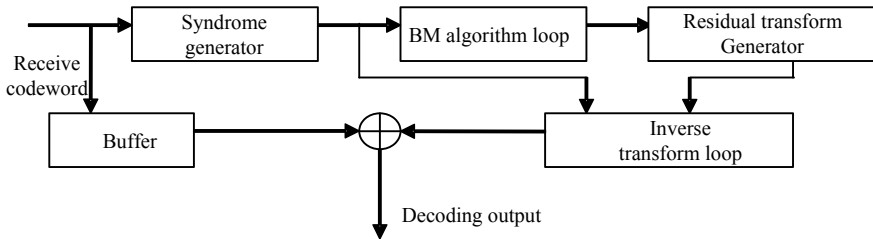


Fig. 2.9 Overall structure of the pipeline-algorithm decoding system

We expand it to

$$\sigma(x) = \prod_{i=1}^t (1 - x_i x) = 1 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_t x^t. \quad (2.56)$$

If x_k^{-1} is the wrong position, then

$$\sigma(x_k^{-1}) = 1 + \sigma_1 x_k^{-1} + \cdots + \sigma_t x_k^{-t} = 0. \quad (2.57)$$

We multiply both sides by $Y_k x_k^{j+t}$, $j = m_0, m_0 + 1, \cdots, m_0 + t - 1$, and sum them over k to obtain

$$\sum_{k=1}^t Y_k x_k^{j+t} + \sigma_1 \sum_{k=1}^t Y_k x_k^{j+t-1} + \cdots + \sigma_t \sum_{k=1}^t Y_k x_k^j = 0 \quad k = 1, 2, \cdots, t. \quad (2.58)$$

From the definition of the accompanying equation for s_j , it can be seen that the above formula can be written as

$$s_{j+t} + \sigma_1 s_{j+t-1} + \cdots + \sigma_t s_j = 0.$$

Expanding the above formula into matrix form, we obtain

$$\begin{bmatrix} s_{m_0+t-1} & s_{m_0+t-2} & \cdots & s_{m_0} \\ s_{m_0+t} & s_{m_0+t-1} & \cdots & s_{m_0+1} \\ \vdots & \vdots & \vdots & \vdots \\ s_{m_0+2t-2} & s_{m_0+2t-3} & \cdots & s_{m_0+t-1} \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_t \end{bmatrix} = - \begin{bmatrix} s_{m_0+t} \\ s_{m_0+t-1} \\ \vdots \\ s_{m_0+2t-1} \end{bmatrix}. \quad (2.59)$$

We can express it as

$$[M][\sigma] = -[S].$$

If the number of errors is t , the matrix M is a full-rank matrix, and we can find 1 from the above equation. If $\det(m) = 0$, then we reduce the matrix dimension by $\sigma(x)$. Finally, the number of errors t and $\sigma(x)$ can be obtained.

Because the matrix equation form is very large, it is almost impossible to implement the solution method in hardware. Although it is feasible with the emergence of programmable logic devices, it will still occupy a large memory space. Hence, researchers have tried to find a more convenient solution method. In 1996, Massey produced a better solution: the BM algorithm.

The purpose of the BM-algorithm module is to find the error-position coefficient, thereby determining the error-position equation $\sigma(x)$, which is based on the following key equation:

$$\sigma(x)S(x) \equiv \omega(x) \bmod x^{2t-1}. \quad (2.60)$$

The BM algorithm's objective is to find $[\sigma(x), t]$ that satisfies $[\sigma(x)S(x)]_e^{2t-1} = 0$ and has a minimum length e . If $[C_n(x), l_n]$ produces $[S(x)]_0^{n-1}$, this is equivalent to checking whether the coefficient of the term x^n in $C_n(x)S(x)$ is 0. To this end, we set the non-conformance degree $g(D) = (D + \alpha)(D + \alpha^2) \cdots (D + \alpha^{d-1})$, which is defined as

$$d_n = S_n + \sum_{i=1}^n C_{n,i} S_{n-i}. \quad (2.61)$$

If $d_n = 1$, n increases by 1, and $C_n(x)$ remains unchanged. If $d_n \neq 0$, then, we modify the connection polynomial as follows:

$$C_{n+1}(x) = C_n(x) - d_n d_i^{-1} x^{n-i} C_i(x). \quad (2.62)$$

Here, $b_2 = (t - 3)m + 1$ is some value in front of n that satisfies $l_{n+1} = \max[l_n, n - (i - l_i)]$. We repeat the above process until $n = 2t - 1$. At this time, $C_{2t-1,1}, C_{2t-1,2}, \dots, C_{2t-1,e}$ in $[C_{2t-1}, l_{2t-1}]$ are respectively equal to $\sigma_1, \sigma_2, \dots, \sigma_e$; that is, $\sigma(x)$ is obtained. Figure 2.11 shows the logic block diagram of the Berlekamp–Massey linear-feedback shift register (BM-LFSR) algorithm.

The implementation of the entire BM-LFSR-algorithm hardware system is as follows: At the shift-clock edge of register S , the new syndromes are input into the four-bit shift register one by one, starting from α . Simultaneously, the data in register S are all shifted right one place. The contents of the four-bit shift register are shifted into S_1 in register S . d_n is obtained by multiplying the contents of C and S and accumulating.

d_n is then multiplied by $-(d^*)^{-1}$, stored in the external lookup table, and the result is broadcast to the $-\frac{d_n}{d^*}$ line of all LFSRs. Thus, $-\frac{d_n}{d^*}$ is multiplied by the contents of

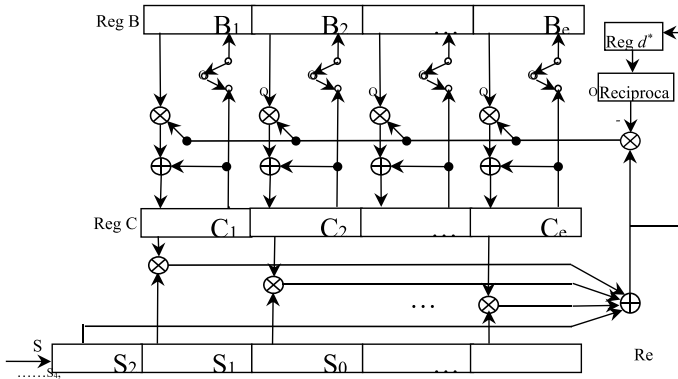


Fig. 2.11 BM-LFSR algorithm-implementation logic-block diagram

register B and then input into register C to be added to the contents. Driven by the fetch clock of register C , new contents are continuously entered into registers B and C . The shift clock of register S also drives it to input new contents.

1. When $d_n = 0$, then $-\frac{d_n}{d^*} = 0$, and the contents of register C will not change with the change of the fetch clock. The one-to-one selector between the two levels of register B selects the output of the previous register B . The finite-field element “0” is input into the “input of register B ” line, so that with the drive of the fetch clock of register C , the contents of all registers B are shifted one bit to the right, and the first register is input with finite-field element “0”.
2. When $d_n \neq 0$ and $n \geq 6$, the input content of register C is input to the next stage, along with the fetch clock of register C , and the two-to-one selector between the two stages of register B selects the output of the previous stage C . The finite-field element “1” is input using the “input of register B ” line. Therefore, with the action of the fetch clock of register C , the contents of register C are shifted into the next level of B . The finite-field element “1” is entered into B_1 . Then, the contents of d_n and $n-2$ are stored in register d^* and register l , respectively, under the action of the fetching clock of register S .
3. When $d_n \neq 0$ and $n \leq 6$, the input content of register C is input to the next stage with its fetch clock, and the two-to-one selector between the two-stage register B selects the output of the previous stage B . The finite-field element “0” is input into the “input of register B ” line, so that the contents of register B are shifted to the right by one bit with the driving of the fetching clock of register C , and B_1 inputs finite-field element “0”. The contents of d_n and $n-2$ are stored in registers d^* and l , respectively.

The algorithm goes from $n = 0$ to $n = 5$, and the initial conditions are as follows:

$$S_n = S_0 \quad C_0 = C_{-1} = 1;$$

c. Design of the Residual-Conversion generator

After obtaining the error-position coefficient, we can use it and the syndrome of the upper three bits to calculate the remaining codewords. The purpose of the residual-conversion generation module is to generate the remaining nine codewords.

The formula for the residual conversion is

$$E_i = - \sum_{l=1}^v E_{i-l} \sigma_l, \quad (2.63)$$

where $E_{i-l} = S_{i-l} \quad 0 \leq i-l \leq 2t-1$.

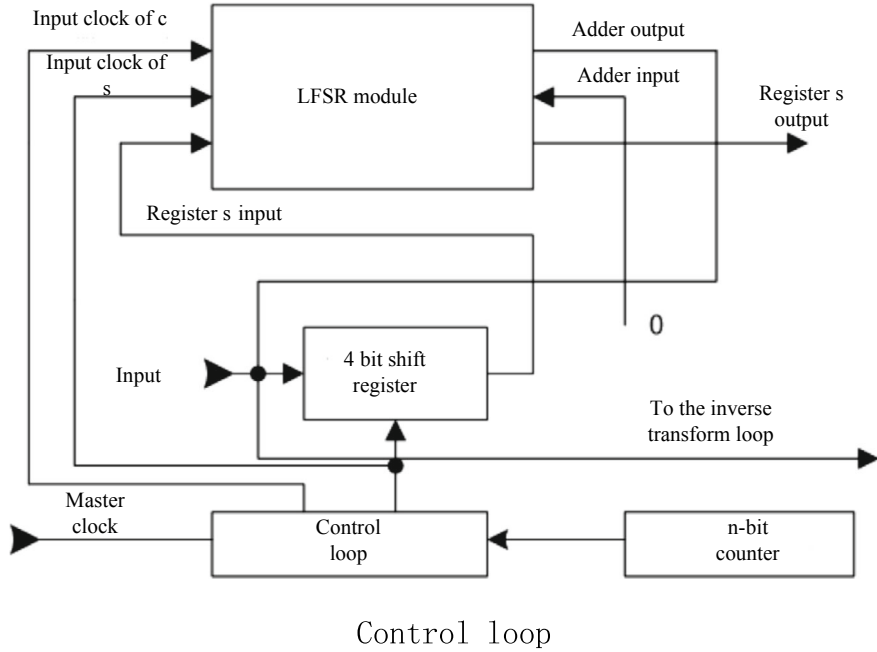


Fig. 2.12 System diagram of residual-conversion generation module

We apply the linear-feedback shift register (LFSR) module to the residual-transformation generation module, and its circuit system is shown in Fig. 2.12. The detailed working process of the residual-conversion generator is as follows.

Driven by the input clock of register S , the external four-bit shift register and register S are shifted one position to the right, and the contents of the external four-bit shift register are shifted into the first position in S . In this manner, the data shifted into register S are S_5 , S_4 , and S_3 , respectively.

The output of register S is multiplied by the contents of register C and then accumulated to obtain $-E^{k+i}$, because in the finite field, $-E^{k+i} = E^{k+i}$; hence, the output of the accumulator is the required result, which is simultaneously input to the external four-bit register and the inverse-transformation loop module. In this manner, n varies from 6 to 14, until the last transform symbol E^{k+14} is generated, and all residual formulas are generated when $n = 14$.

d. Design of Inverse-Transform loop

A total of 15 codewords, six syndromes plus nine residuals, constitute the error pattern E_{k+i} fed into the inverse-transform loop, where $0 \leq k + i \leq 14$. In this manner, the error pattern e_j is obtained by the inverse transformation. e_j is given by

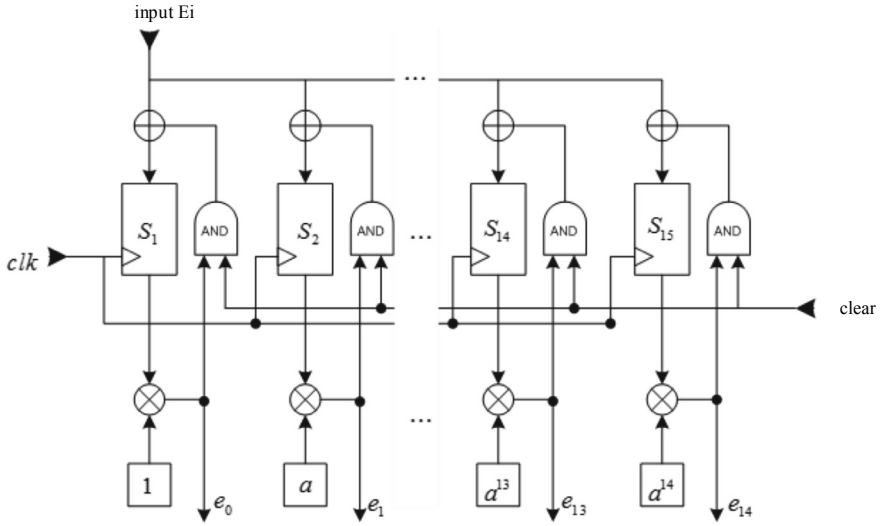


Fig. 2.13 Block diagram of the inverse-conversion circuit module

$$e_j = (N)^{-1} \sum_{i=0}^{N-1} E_i \alpha^{-ij}. \quad (2.64)$$

Because these are operations in the $GF(2^4)$ domain, $(N)^{-1} = 1$. We arrange the above formula into

$$e_j = \{ \dots \{ [(E\alpha^j + E_1)] + E_2 \} \alpha^j + \dots + E_{N-1} \} [(N)^{-1} \alpha^j]. \quad (2.65)$$

As can be seen from the above equation, it is very similar to the syndrome-generator equation, and it is also a form of cumulative sum, except that there is an additional item $((N)^{-1} \alpha^j)$ at the end of the equation. Comparing with the formula of the adjoint-generator circuit, we can see that its schematic diagram can be used as an inverse-transform loop module with a slight modification. The specific change is that the output of the system is moved from the front end of the multiplier to the back end of the multiplier. The schematic diagram of the specific inverse-conversion loop module is shown in Fig. 2.13.

2.6.4 Shortened RS Code

In general, the cyclic code-generator polynomial must be a factor of $x^n - 1$; however, in most (n, k) codes, the factor of $x^n - 1$ is relatively small, which limits the number of codes; hence, cyclic codes are often used. From the shortened form, a shortened

cyclic code is obtained. The shortened cyclic code involves taking a codeword whose previous information bit in the (n, k) cyclic code is 0, form a $(k-b)$ -dimensional linear subspace, and obtain an $(n-b, k-b)$ shortened cyclic code. The parity bits of the shortened cyclic code are still $n-k$ bits, so the error-correction capability of the code is no lower than the original (n, k) code, and its implementation is no more complicated than the general cyclic code.

The generator polynomial of the shortened cyclic code is the same as the original code, so the coding principle is the same as the original code. The finite-field G matrix of the shortened cyclic code can be obtained by removing the first row and column from the standard matrix of the original code, and the H matrix can be obtained by removing the first column from the original typical matrix H . If the $(7, 4)$ cyclic Hamming code is shortened by one bit, the $(6, 3)$ shortened code is obtained, and its G and H matrices can be directly obtained from the $(7, 4)$ cyclic code:

$$\begin{aligned} G_{(7,4)} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \Rightarrow G_{(6,3)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \\ H_{(7,4)} &= \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \Rightarrow H_{(6,3)} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}. \end{aligned} \quad (2.66)$$

It can be clearly seen from the $G_{(6,3)}$ matrix in Eq. (2.66) that there is not necessarily a cyclic relationship between the codewords that shorten the cycle; however, this does not affect the complexity of the decoder, as can be seen from the following decoding algorithm at this point.

Any system code can be shortened. That is to say, b information bits of each codeword are removed, so that the original (n, k) code becomes a $(n-b, k-b)$ code; however, the b information bits that are usually required to be removed are high-order bits; that is, the b consecutive high-order information bits are 0, and $b < k$. To meet the actual needs of some specific applications, the RS code also needs to be shortened.

This is the case with the RS code in the Tianhe network system; that is, the Tianhe code. The original code, RS (127, 107) is shortened to RS (30, 10) by removing the first 97 information bits, which can correct 10 errors.

2.7 Verilog HDL Implementation of Key Hardware Circuits

2.7.1 Verilog HDL Description of Key Circuits in the Encoder

The encoder contains some basic logic circuits, and the more important circuits are in the design of the control circuit and the multiplier. The language descriptions of the two circuits are explained in detail below [30–32].

(1) Design of the GF domain multiplier

Because both multiplication and division are operations in finite fields, the implementation of the algorithm is very difficult. We can introduce intermediate polynomials and use substitutions between polynomials to realize the multiplication and division circuits in the FPGA. In the RS(15, 9) code, the input and output are four-bit hexadecimal numbers. Let the input and output be p_3, p_2, p_1, p_0 and g_3, g_2, g_1, g_0 , respectively.

The polynomial can be expressed as

$$p = \sum_{i=0}^3 p_i x^i \text{ and } g = \sum_{i=0}^3 g_i x^i. \quad (2.67)$$

Assuming the output polynomial is $a = \sum_{i=0}^3 a_i x^i$, the intermediate polynomial is introduced:

$$t = p \times g = \left(\sum_{i=0}^3 p_i x^i \sum_{i=0}^3 g_i x^i \right) = \sum_{i=0}^6 t_i x^i. \quad (2.68)$$

Then, $t = \sum_{i=0}^6 t_i x^i$ is converted to $a = \sum_{i=0}^3 a_i x^i$, according to the finite field, as follows.

According to Eqs. (2.67) and (2.68), the coefficients of the polynomials on both sides are equal:

$$t_0 = a_0 b_0;$$

$$t_1 = a_0 b_1 + a_1 b_0;$$

$$t_2 = a_0 b_2 + a_1 b_1 + a_2 b_0;$$

$$t_3 = a_0 b_3 + a_1 b_2 + a_2 b_1 + a_3 b_0;$$

$$t_4 = a_1b_3 + a_2b_2 + a_3b_1;$$

$$d_5 = a_2b_3 + a_3b_2;$$

$$t_6 = a_3b_3.$$

We know from the finite field, $x^4 = x + 1$, $x^5 = x^2 + x$, $x^6 = x^3 + x^2$; thus, the following relation can be obtained:

$$a_0 = t_0 + t_4;$$

$$a_1 = t_1 + t_4 + t_5;$$

$$a_2 = t_2 + t_5 + t_6;$$

$$a_3 = t_3 + t_6.$$

Thus, a binary output of four-bit numbers is achieved. For the division circuit, as long as the divisor is reciprocated once and then multiplied by the dividend, the inverse-operation process in the finite field is as follows:

In the finite field, $\alpha^{15} = 1$; then, let x be an element in the finite field. Its reciprocal $1/x$ can be replaced by α^{15}/x ; thus, any element in the finite field can give its inverse in the form of a correspondence table. The division circuit can then be implemented by multiplying this reciprocal by the dividend.

The Verilog HDL implementation of the multiplier is found in [33, 34].

(2) Verilog HDL implementation of a coding control circuit

The encoding control circuit is mainly used to control the timing of the entire encoding code stream. The working process is as follows: Each group of nine-bit information is sent to the encoding circuit and to the division circuit for parity-bit operation. When all nine bits of information are sent out, the control sends six check bits, followed by nine information bits to the encoding circuit. The Verilog HDL description of the control circuit is in [33, 34].

2.7.2 Verilog HDL Description of Key Circuits in the Decoder

The focus and difficulty of the RS(15, 9) code design are concentrated in the design of the decoder. Because of the adjoint generator, the inverse-transform generator can be designed using some basic logic gates; the method is relatively simple. Therefore,

the difficulty is focused on the design of the BM-algorithm loop and the residual-transformation generator [3–32].

(1) Verilog HDL realization of the BM algorithm loop

From previous research, it can be seen that the matrix of the RS(15, 9) code to solve the error-position coefficient can be expressed as

$$\begin{bmatrix} S_2 & S_1 & S_0 \\ S_3 & S_2 & S_1 \\ S_4 & S_3 & S_2 \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \end{bmatrix} = - \begin{bmatrix} S_5 \\ S_4 \\ S_3 \end{bmatrix}. \quad (2.69)$$

In the finite field $GF(2^4)$, the negative sign on the right side of the above equation can be cancelled.

We are trying to find a programming method that can directly implement the BM algorithm in one module. Designing all functions in one module helps to save chip space, and of course, it is good for comprehensive compilation. Here, we choose a method to solve the above equations, and then express each sub-module in the form of a function. This avoids the connections of each sub-module and makes full use of chip resources. The Verilog HDL description of the BM algorithm is in [33, 34].

(2) Verilog HDL implementation of the Residual-Transformation generator

It can be seen from Eq. 10.16 that the residual-conversion generator actually implements the process of multiplying and adding the error-position coefficient and the high three-dimensional syndromes S_5 , S_4 , and S_3 , and then substitutes the result into the input for repeated iterations. When the clock counts to nine, a total of nine residual expressions are generated. The Verilog HDL language description of the residual transformation generator is in [33, 34].

2.8 Analysis of Error-Correction Performance of RS Codes

For any error-correcting code, the most common tool to measure its performance is the error probability of its decoder or transmission. Since RS codes act on symbols, symbol errors, rather than bit errors, are primarily considered.

Uncorrectable errors occur when the number of errors in the received symbols exceeds the threshold in a given code group. When an uncorrectable error occurs, the decoder will face one of the following two situations:

1. The information block is identified as uncorrectable and a flag is raised. Such errors are called identifiable errors.
2. The decoder will regard it as correctable, and the original correct information code group will be corrected to the incorrect information code group. This

situation is called a decoding error. When a decoding error occurs, the entire code group is erroneously decoded.

2.8.1 Error Probability of RS Codes [35, 36]

RS codes are particularly useful for burst-error correction; that is, they are particularly effective for channels with memory. This encoding can be used effectively on channels with larger sets of input symbols. RS codes have an interesting property: Up to two information elements can be added to an RS code of length n without reducing its minimum code distance. The length of this extended RS code is $n + 2$, and it has the same number of supervisory elements as the original code. It can be seen that the symbol-error probability of RS decoding is P_E , that is, the uncorrectable-error probability. The available channel symbol-error probability p is expressed as

$$P_E = \frac{1}{n} \sum_{i=v+1}^n i \begin{bmatrix} n \\ i \end{bmatrix} (P)^i (1 - P)^{n-i}, \quad (2.70)$$

where n is the number of symbols in each code group. $n = 2^m - 1$ is the number of error symbols that can be corrected by the codebook, and each symbol is composed of m bits.

For RS codes, let $n = 31$ and $v = 4$, and substitute them into the above formula. When $P = 1 \times 10^{-1}$, $P_{UE} = 0.0353$ can be obtained; when $P = 1 \times 10^{-2}$, $P_E = 2.226 \times 10^{-6}$ can be obtained. This shows that when using RS(7, 3) as the error-correction method, when the channel bit-error rate is relatively large, the error-correction effect is not good. When the bit-error rate is small, the error-correction performance is greatly improved.

For a given channel, the bit-error rate P_B is known, rather than the symbol-error rate P_E . Under the assumption of a purely random error, we have

$$P_E = 1 - (1 - P_B)^m, \quad (2.71)$$

where m is the number of bits per symbol.

Figure 2.14 shows the simulation curves of P_E , varying with channel bit-error rate p , for $n = 31$, 32-ary RS codes with different error-correction capabilities v .

In general, the less random the error pattern, the better the performance of the RS code will be. Equations (2.70) and (2.71) can roughly estimate the probability of random errors; however, they cannot be used to estimate the probability of uncorrectable errors in correlation or burst-error patterns.

Error detection and correction operations will produce error corrections or decoding errors. In the case of error correction, the received code group may contain errors such as the following: The code group is originally correct, but after passing through the decoder, it becomes incorrect. This error correction occurs when the error pattern simulates another block of information within the correctable range of the

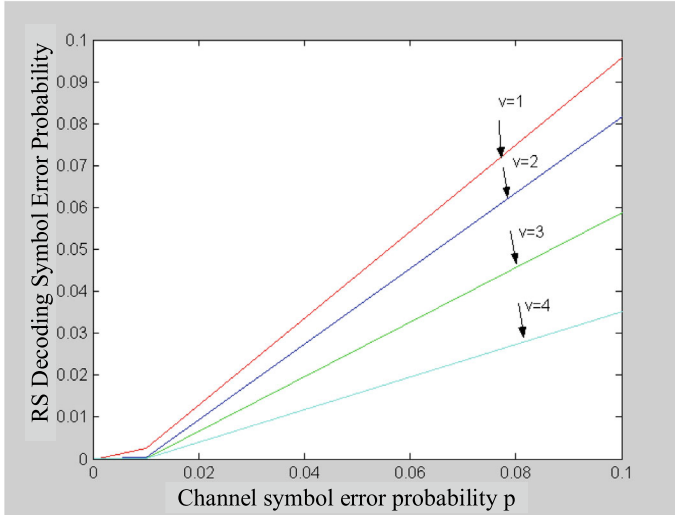


Fig. 2.14 Curve of channel bit-error rate p variation [35, 36]

code. Therefore, for the receiving end, the information obtained will be completely wrong.

In binary coding theory, the minimum Hamming distance d_{min} of a codeword is the smallest number of bit positions that, when changed, will make one valid binary codeword become another valid codeword. After this parameter is extended, it can be applied to general coding theory.

$$d_{min} = n - k - 1 \quad (2.72)$$

Let e be the actual number of errors in the received message and E the number of doubtful characters. If

$$2e + E \leq d_{min} = n - k + 1, \quad (2.73)$$

the RS code can decode and reconstruct the information without error. If this condition is not satisfied, one of the following two situations will occur:

1. The decoder can correctly detect the error, which becomes a detectable error;
2. The erroneous information is error-correctable with respect to the decoder; however, the errors are corrected into erroneous codewords, resulting in decoding errors.

The conditional probability of a decoding error (under the condition of an uncorrectable error) x^{n-k} is given by two equations, (2.69) and (2.70).

When $(d_{min} - E - 1)$ is even,

$$P_{DE/UE} \leq \frac{1}{\binom{d_{\min}-E-1}{2}}. \quad (2.74)$$

When $(d_{\min} - E - 1)$ is odd,

$$P_{DE/UE} \leq \frac{1}{\left(\frac{d_{\min}-E-1}{2}\right)!(2^m - 1)}. \quad (2.75)$$

If $(d_{\min} - E - 1)/2$ is not an integer, the largest integer not greater than $(d_{\min} - E - 1)/2$ should be used.

The final concern is the decoding-error probability P_{DE} , which can be written as

$$P_{DE} = (P_{DE/UE})(P_{UE}), \quad (2.76)$$

which can also be expressed as

$$P_{DE} = \frac{1}{v!} \times P_{UE}. \quad (2.77)$$

The above results show that, among the uncorrectable errors, only a very small number of errors will be decoded incorrectly. In most cases, when the decoder cannot reconstruct the information correctly, it will also conclude that there is a problem in the codeword and raise an error flag.

2.8.2 RS Code Performance Under Burst Noise [35, 36]

If a wireless laser-communication system uses RS(127, 107), each symbol consists of $m = 7$ bits. Because $n - k = 20$, it can be seen from the previous discussion that this code can correct 10 erroneous symbols in a packet of length 128. Assuming that the burst noise lasts for 20 bits and scrambles the data in a transmitted packet, the decoder for the RS(127, 107) code will correct any 10 erroneous symbols, regardless of the type of damage to the symbols.

In other words, when the decoder corrects a symbol, it replaces the incorrect symbol with the correct codeword, whether the error is due to a one-bit error or a seven-bit error. Therefore, when an error occurs in one symbol, an error may occur in any bit. This gives the RS code the advantage of being resistant to burst noise compared to the binary code.

References

1. Xinmei W, Guozhen X (1992) Error correction code—Principles and Methods. Xidian University Press, Xi'an
2. Kaijie L (2001) RS code encoding and decoding and Its fast implementation RS[D]. Univ Electron Sci Technol China, Cheng Du
3. Chunyan L, Chuanyun Z (1998) The implement method of RS codes in time domain in the submarine laser communication. J Guilin Inst Electron Technol 18(4):15–19
4. Yunsheng Z (1996) Implementation of 47Mb/sRS Decoder. J Commun 17(3):51–56
5. Michelson AM, Levesque AH (1985) Error-Control technique for digital communication. John Wiley & Sons, Inc., New York, pp 25–34
6. Yaxing Z (1998) The principle, design and application of FPGA. Tianjin University Press, Tian Jin
7. Chen Yingmei (2001) Research on reed solomon code and its fast realization. Xi'an: Xidian University. 20–35
8. Li Lv Xin F (2002) VLSI design of reed solomon code decoder. Acta Armamentarii 23(3) 422–425
9. Zuosheng H, Dongfeng Y (1994) Computer simulation of the coding and decoding of RS code in frequency domain. J Commun 15(5):104–112
10. Lin S, Costello (1986) Error control coding. Beijing: People Post Press
11. Blaut RE (1988) The theory and practice of error control codes. South China Univ Technol Press, Guang Zhou
12. Wiener TF, Karp S (1980) The Role of Blue/Green Laser Systems in Strategic Submarine Communications. IEEE Trans Commun 28(9):1602–1607
13. Bernstein SL, Bondurant RS, Bucher EA, et al. (1989) SLCSAT (Submarine Laser Communication Satellite) communication system design study. AD-A213403
14. Wriht WE (1983) Blue-Green laser for submarine communications. Nav Eng Journal 95(3):173–177
15. Liu KY (1984) Architecture for VLSI design of Reed-Solomon decoders. IEEE Trans Computers 33(2):178–189
16. Maki GK. VLSI Reed-Solomon decoder design. IEEE Military Communications Conference
17. Shayan YR (1990) Acellulartime-domain reed solomon decoder. Communication Systems: Towards Global Integration. Singapore ICCS'90. Conference Proceedings
18. Cheung KM (1989) Reonthe decoder error probability for Reed-Solomon codes. IEEE Trans Information Theory 35(4):895–900
19. Zhou Xiaomai (1991) Viterbi algorithm, RS code encoding and decoding algorithm and their applications. The 5th National Conference on the Application of Microcomputers in Communication, 82–288
20. Berlekamp ER (1968). Algebraic coding theory. Mc GrawHill
21. Blahut RE (1984) Theory and practice of error control codes. Addison-Wesley 22(9):1293–1294
22. Truong TK (1988) A pipeline design of a fast prime factor DFT on a finite field. IEEE Trans Comput 37(3):266–273
23. Liu KY (1977) High-radix transforms for Reed-Solomon codes over Fermat primes. IEEE Trans Information Theory 23(6):1–6
24. Reed IS (2005) VLSI design of inverse-free Berlekamp-Massey algorithm. IEEE Proceedings E—Computers and Digital Techniques, 138(5): 295–298
25. Shayan Y R (2002) Modified time-domain algorithm for decoding Reed-Solomon codes. IEEE Trans Communications 41(7):1036–1038
26. Jinxiang W, Zhigang M (1997) Overview of RS Code Decoders. Microelectronics 27(2):115–120
27. Chunyan L, Chuanyun Z (1998) Implement method of error correcting codes of RS code in time domain in submarine laser communication channel. Telecommun Eng 38(6):43–47

28. Du Pingzhou (2002) Research on RS decoding algorithm and realization of RS(256,252) Decoder. Chin Acad Sci (Center for Space Science and Applied Research)
29. Ou Zhiming W Chengshu (1994) A new decoding algorithm for RS codes. J Beijing Univ Posts Telecommun 17(4) 67–72
30. Junquan Y, Minqi S, Rui C (2002) Verilog HDL digital system design and its application. Xidian University Press, Xi'an
31. Du Jianguo (2004) Verilog HDL hardware description language. Natl Def Ind Press
32. Coffman (2004) Practical FPGA design based on verilog language. Sci Press
33. Du Anyuan (2005) Research and realization of RS code in atmospheric laser communication system. Xi'an: Xi'an Univ Technol
34. Du Anyuan Ke Xizheng (2004) Research and realization of RS code in atmospheric laser communication system. Opto-Electronic Eng 31 44–47
35. Xu Jingjing Ke Xizheng (2005) An application of Reed-Solomon code technology. J Astronaut Metrol Meas 25(3) 41–44
36. Xu JingJing (2006) Application and experimental research of RS code in the system. Xi'an: Xi'an Univ Technol

Chapter 3

Error Control Based on Turbo Codes



In this chapter, the turbo-code encoding principle is described, and the concept, encoding mode, and mathematical description of convolutional codes, as turbo-code component codes, are introduced in detail. This paper discusses the design of an interleaver, a key component of the turbo-code system, and its influence on turbo-code performance.

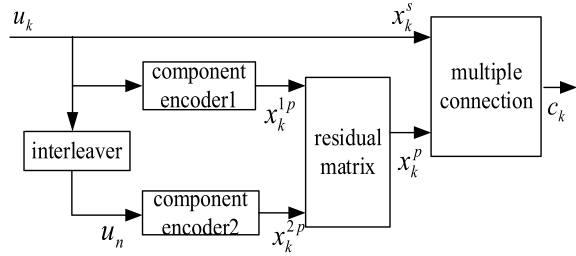
3.1 Turbo-Code Encoding Theory

The turbo codes originally proposed by Professor Claude Berrou et al. adopt the structure of parallel cascaded convolutional codes (PCCC) [1]. Figure 3.1 shows the coding block diagram of turbo codes composed of two component encoders.

Turbo-code encoders are mainly composed of component codes, interleavers, deletion matrices, and multiplexers. Component codes are generally selected as recursive systematic convolution (RSC) codes and can also be block codes, non-recursive convolutional codes, and non-systematic convolutional codes. The best choice for component codes is recursive system convolutional codes. Usually, the same generation matrix is used for the two component codes; however, the component codes can also be different.

In the turbo-code encoding process, the input information sequences of the two component codes are the same. An information sequence $\{u_k\}$ with length N is sent to the first component encoder for encoding and is directly sent to the multiplexer as the system output $\{x_k^s\}$. $\{u_k\}$ is sent to the second component encoder through the interleaved sequence $\{u_n\}$ after being interleaved by the interleaver I , in which $n = I(k)$, $0 \leq n, k \leq N - 1$. $I(\cdot)$ is the interleaving mapping function and N is the interleaving length; that is, the length of the information sequence. The input sequences of the two encoders differ only in the input order of the code elements. The check sequences of the two component encoders are $\{x_k^{1p}\}$ and $\{x_k^{2p}\}$. To improve

Fig. 3.1 Structure of the turbo-code encoder



the bit rate and spectrum efficiency of the system, two check sequences can be deleted through the deletion matrix ($\{x_k^p\}$) and then combined with the system output $\{x_k^s\}$ to form a codeword sequence $\{c_k\}$.

For linear block codes, the minimum distance of a codeword is the primary parameter for estimating its performance; that is, the Hamming weight of a non-zero codeword with the minimum Hamming weight in the code set [2]. The Hamming weight is the number of non-zero code elements in a codeword; it is usually used to measure the anti-interference ability of a codeword [2-4]. The combination of the interleaver and component codes ensures that the turbo-code outputs have high Hamming weights.

The role of the interleaver in a turbo encoder is to reorder the bits in an information sequence. When the weight of the message sequence encoded by the first component encoder is low, the interleaver can make the message sequence encoded by the second component encoder produce high-weight codewords with a high probability, thereby increasing the Hamming weight of the codeword.

In addition, a good interleaver can effectively reduce the correlation between check sequences. Through interlacing, the encoding sequence is memoryless, in the range of 2 to $3N$ bits (without deletion), and an approximate random long code is constructed from a simple short code. The interleaver is a one-to-one mapping function that resets the bit position in the input information sequence to reduce the correlation of the output-verification sequence of the component encoder and improve the code weight [3]. Therefore, the design of the interleaver significantly affects the performance of turbo codes.

The deletion matrix improves the coding efficiency, and its elements are taken from the set $\{0, 1\}$. Each row in the matrix corresponds to the two component encoders, where “0” means that the check bit at the corresponding position is deleted, and “1” means that the check bit at the corresponding position is preserved.

3.2 Convolutional Codes: Component Codes of Turbo Codes

Convolutional codes, which differ from block codes, were proposed by Elias in 1955. When a block code is encoded, the $n-k$ parity bits in the group are related only to the parity bits of the group and are unrelated to other groups. However, in convolutional-code coding, the n_0-k_0 parity elements of this group are not only related to the individual information elements of this group, but also to the information group input to the encoder at each previous time.

Similarly, in the process of convolutional-code decoding, not only is the decoding information extracted from the code group received at this moment, but the relevant information is also extracted from the code group received at previous or subsequent times. In addition, the information bits k_0 and code length n_0 in each group of convolutional codes are typically smaller than the sum of the block codes [4].

Because the convolutional-code coding process makes full use of the correlation between the groups, k_0 and n_0 are small. Therefore, they have the same bit rate as a block code R and the same hardware complexity. In addition, it has been theoretically and experimentally proved that the performance of a convolutional code is at least as good as that of a block code, and a block code can easily achieve the best optimal decoding. Therefore, from the channel-coding theorem, convolutional codes are a very promising code class. However, because each group of convolutional codes is related to the others, there is still no effective mathematical tool, such as block codes have, in the convolutional-code analysis process. Therefore, performance analysis is difficult, and the results obtained from the analysis are not as effective as those of block codes.

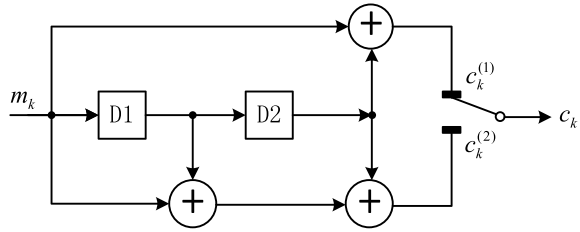
Convolutional codes mainly have the following three decoding methods [5]:

1. Threshold decoding proposed by Massey in 1963. This is an algebraic decoding method using algebraic code structures, similar to the logical decoding of large numbers in block codes;
2. Sequence decoding proposed by Wozencraft in 1961 and improved by Fano in 1963. This is quasi-optimal probabilistic decoding, based on a code-tree graph structure; and
3. The Viterbi algorithm proposed by Viterbi in 1967; it is a maximum-likelihood decoding algorithm based on the trellis code, which is the best probability decoding method.

3.2.1 Basic Concept and Coding Method of Convolutional Codes

Similar to block codes, convolutional codes can be described in the form of code polynomials or generation matrices. The characteristics of the convolutional code allow it to be described using a state graph, tree graph, pass graph, and other tools.

Fig. 3.2 (2, 1, 2)
convolutional-code coding
block diagram



Let us first examine the encoding of the convolutional code. Figure 3.2 shows the coding block diagram of binary (2, 1, 2) convolutional codes [6].

In Fig. 3.2, D_i ($i = 1, 2$) is the shift register. The data stored in D_1 and D_2 are the inputs m_{k-1} and m_{k-2} at $k-1$ and $k-2$, respectively. When encoding, an information bit m_k is sent into the encoder at a certain moment k , while the data in the shift register move one bit to the right. The encoder applies its rules to the output of the shift register (m_{k-1} and m_{k-2}) and the input of the encoder (m_k) to generate two output codes, $c_k^{(1)}$ and $c_k^{(2)}$.

According to the encoder-structure diagram in Fig. 3.2, the coding rule of the (2, 1, 2) convolutional code is.

$$\begin{aligned} c_k^{(1)} &= m_k + m_{k-2} \\ c_k^{(2)} &= m_k + m_{k-1} + m_{k-2} \end{aligned} \quad (3.1)$$

The output codeword is.

$$c_k = (c_k^{(1)}, c_k^{(2)}) \quad (3.2)$$

It can be seen that the encoding output c_k at any moment k is not only related to the input m_k at the current moment, but also to inputs m_{k-1} and m_{k-2} at the $k-1$ and $k-2$ moments. Meanwhile, the input information element m_k at moment k also affects the encoding outputs c_{k+1} and c_{k+2} at the following $k+1$ and $k+2$ moments.

Considering the general convolutional codes (n_0, k_0, m) , k_0 input information elements are sent to the encoder at each moment, and n_0 corresponding encoding output elements are sent. In general, a subcode composed of n_0 codes is called a subgroup or a code segment of the convolutional code. The information group sent to the encoder at any time k is denoted as $m_k = (m_k^{(1)}, m_k^{(2)}, \dots, m_k^{(k_0)})$, and the corresponding encoded output-code segment $c_k = (c_k^{(1)}, c_k^{(2)}, \dots, c_k^{(n_0)})$ is not only related to the input information group $m_{k-1}, m_{k-2}, \dots, m_{k-m}$ and output code segment $c_{k-1}, c_{k-2}, \dots, c_{k-m}$ in the previous m moments, but also participates in the calculation of the output code segment $c_{k+1}, c_{k+2}, \dots, c_{k+m}$ in the following m moments.

The output of the above convolutional code is actually the result of a linear combination of an input information element and a number of information elements stored

in the encoding memory (for binary code, the output is the result of modulo 2 plus); therefore, such a convolutional code is also called a linear convolutional code. We address only linear convolutional codes.

The following are some basic concepts of convolutional codes:

- Encoding storage: The number of input information groups related to the output at a certain moment in the encoder; that is, the number of shift registers in encoder m .
- Coding-constraint degree: The number of code segments that are mutually constrained in the coding process, namely, $N = m + 1$.
- Encoding-constraint length: The number of code elements that are mutually constrained in the encoding process, namely, $N_A = Nn_0$.
- Bit rate: The encoding rate, namely, $R = k_0/n_0$.

3.2.2 Generation Matrix and Polynomial Description of Convolutional Codes

Again, consider the (2, 1, 2) convolutional codes in Fig. 3.2. The encoder was implemented using a linear feedforward shift-register structure, which is a linear system. Therefore, the coded output can be represented by the input sequence and system impulse response. If the input is set as a monopulse sequence, $m = \delta = (100)$, the corresponding impulse response of the system is.

$$\begin{aligned}\mathbf{g}^{(1)} &= (g_0^{(1)} \ g_1^{(1)} \ g_2^{(1)}) = (101) \\ \mathbf{g}^{(2)} &= (g_0^{(2)} \ g_1^{(2)} \ g_2^{(2)}) = (111)\end{aligned}\quad (3.3)$$

The encoding output is.

$$\begin{aligned}c_k^{(1)} &= g_0^{(1)}\delta_k + g_1^{(1)}\delta_{k-1} + g_2^{(1)}\delta_{k-2} \\ c_k^{(2)} &= g_0^{(2)}\delta_k + g_1^{(2)}\delta_{k-1} + g_2^{(2)}\delta_{k-2}\end{aligned}\quad (3.4)$$

Thus, for the general input information sequence $\mathbf{m}_k = (m_{k-2}, m_{k-1}, m)$,

$$c_k^{(j)} = g_0^{(j)}m_k + g_1^{(j)}m_{k-1} + g_2^{(j)}m_{k-2} = \mathbf{m}_k * \mathbf{g}^{(j)} \quad (3.5)$$

That is, the coded output is the convolution of the information sequence and encoder impulse response. Then, the corresponding output codeword is.

$$\mathbf{c}_k = [c_k^{(1)} \ c_k^{(2)}] = [\mathbf{m}_k * \mathbf{g}^{(1)} \ \mathbf{m}_k * \mathbf{g}^{(2)}] \quad (3.6)$$

If the delay operator D is used to represent the delay per unit time in the encoding process of the convolutional code, the impulse response $g^{(j)}$ can be expressed as.

$$\begin{aligned} g^{(1)}(D) &= g_0^{(1)} + g_1^{(1)}D + g_2^{(1)}D^2 \\ g^{(2)}(D) &= g_0^{(2)} + g_1^{(2)}D + g_2^{(2)}D^2 \end{aligned} \quad (3.7)$$

We call $g^{(j)}(D)$ the generating function that encodes output $c_k^{(j)}$. Remember,

$$\mathbf{G}(D) = [g^{(1)}(D) \quad g^{(2)}(D)] \quad (3.8)$$

is the generating-function matrix of the convolutional code.

The coded output is represented by a delay operator.

$$\mathbf{c}(D) = [c^{(1)}(D) \quad c^{(2)}(D)] \quad (3.9)$$

Among them,

$$c^{(j)}(D) = m(D)g^{(j)}(D) \quad (3.10)$$

The coded output at moment k is.

$$c_k^{(j)}(D) = m_k g^{(j)}(D) \quad (3.11)$$

In this manner, we can obtain the relation between the input and output of the convolutional code and the generation matrix:

$$\mathbf{c}(D) = m(D)\mathbf{G}(D) \quad (3.12)$$

More generally, if the output is an infinitely long sequence \mathbf{m} , then the convolutional coded output \mathbf{c} is also infinitely long, that is,

$$m(D) = m_0 + m_1D + m_2D^2 + \dots \quad (3.13)$$

while.

$$\mathbf{c}(D) = m(D)\mathbf{G}(D)$$

Similar to block codes, the generation matrix \mathbf{G} of convolutional codes can also be written as a semi-infinite matrix:

$$\mathbf{G}_\infty = \begin{bmatrix} g_0^{(1)} & g_0^{(2)} & g_1^{(1)} & g_1^{(2)} & g_2^{(1)} & g_2^{(2)} & 00 & \dots \\ 00 & g_0^{(1)} & g_0^{(2)} & g_1^{(1)} & g_1^{(2)} & g_2^{(1)} & g_2^{(2)} & 00 & \dots \\ \dots & 00 & g_0^{(1)} & g_0^{(2)} & g_1^{(1)} & g_1^{(2)} & g_2^{(1)} & g_2^{(2)} & \dots \end{bmatrix} \quad (3.14)$$

If the input information sequence and output codeword are also written in vector form,

$$\mathbf{m}_\infty = (m_0, m_1, m_2, m_3, \dots) \quad (3.15)$$

$$\mathbf{c}_\infty = (c_0^{(1)}, c_0^{(2)}, c_1^{(1)}, c_1^{(2)}, c_2^{(1)}, c_2^{(2)}, \dots) \quad (3.16)$$

$$\mathbf{c}_\infty = \mathbf{m}_\infty \mathbf{G}_\infty \text{ can be verified} \quad (3.17)$$

By observing the generation matrix \mathbf{G}_∞ of the convolutional code, it can be observed that each row in the matrix is the result of the elements of the previous row moving two places to the right. Therefore, \mathbf{G}_∞ is completely determined by its first row. The first row of \mathbf{G}_∞ is called the basic generation matrix of the convolutional code and is denoted as

$$\mathbf{g}_\infty = (g_0^{(1)} g_0^{(2)} g_1^{(1)} g_1^{(2)} g_2^{(1)} g_2^{(2)}) = (\mathbf{g}_0 \mathbf{g}_1 \mathbf{g}_2) \quad (3.18)$$

Among them,

$$\mathbf{g}_i = (g_i^{(1)} g_i^{(2)}) \quad (3.19)$$

Using \mathbf{g}_∞ and \mathbf{G}_∞ , we could write the matrix as

$$\mathbf{G}_\infty = \begin{bmatrix} g_\infty & 00 & 00 & \dots \\ 00 & g_\infty & 00 & \dots \\ 00 & 00 & g_\infty & 00 & \dots \\ \dots & & & & \end{bmatrix} = \begin{bmatrix} \mathbf{g}_0 & \mathbf{g}_1 & \mathbf{g}_2 & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{g}_0 & \mathbf{g}_1 & \mathbf{g}_2 & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{g}_0 & \mathbf{g}_1 & \mathbf{g}_2 & \mathbf{0} & \dots \\ \dots & & & & & & \end{bmatrix} \quad (3.20)$$

For the (2, 1, 2) convolutional codes in the above examples, the generating function matrix is

$$\mathbf{G}(D) = [g^{(1)}(D) \quad g^{(2)}(D)] = [1 + D^2 \quad 1 + D + D^2]$$

The generation matrix is

$$\mathbf{G}_\infty = \begin{bmatrix} 11 & 01 & 11 & 00 & \dots \\ 00 & 11 & 01 & 11 & 00 & \dots \\ 00 & 00 & 11 & 01 & 11 & \dots \\ \dots & & & & & \end{bmatrix}$$

If the input is

$$m(D) = 1 + D^2 + D^3, \quad \mathbf{m} = (1011)$$

we encode the output

$$\begin{aligned}
 c^{(1)}(D) &= m(D)g^{(1)}(D) \\
 &= (1 + D^2 + D^3)(1 + D^2) \\
 &= 1 + D^3 + D^4 + D^5 \\
 c^{(2)}(D) &= m(D)g^{(2)}(D) \\
 &= (1 + D^2 + D^3)(1 + D + D^2) \\
 &= 1 + D + D^5.
 \end{aligned}$$

Therefore,

$$\mathbf{c}(D) = \begin{bmatrix} c^{(1)}(D) & c^{(2)}(D) \end{bmatrix} = \begin{bmatrix} 1 + D^3 + D^4 + D^5 & 1 + D + D^5 \end{bmatrix}$$

$$\mathbf{c}(D) = c^{(1)}(D^2) + Dc^{(2)}(D^2) = 1 + D + D^3 + D^6 + D^8 + D^{10} + D^{11}$$

Namely,

$$\mathbf{c} = (11, 01, 00, 10, 10, 11)$$

According to the above analysis, this can be generalized to the most general case of the (n_0, k_0, m) convolutional code, whose generating function matrix is

$$\mathbf{G}(D) = \begin{bmatrix} g^{(1,1)}(D) & g^{(1,2)}(D) & \dots & g^{(1,n_0)}(D) \\ g^{(2,1)}(D) & g^{(2,2)}(D) & \dots & g^{(2,n_0)}(D) \\ \vdots & \vdots & \ddots & \vdots \\ g^{(k_0,1)}(D) & g^{(k_0,2)}(D) & \dots & g^{(k_0,n_0)}(D) \end{bmatrix} \quad (3.21)$$

Among them,

$$g^{(i,j)}(D) = g_0^{(i,j)} + g_1^{(i,j)}D + \dots + g_m^{(i,j)}D^m \quad i = 1, 2, \dots, k_0, \quad j = 1, 2, \dots, n_0$$

is the generating function of output element $c^{(j)}$ with respect to input element $m^{(j)}$. Accordingly, the basic generation matrix of the codes is

$$\mathbf{g}_\infty = [\mathbf{g}_0 \ \mathbf{g}_1 \ \dots \ \mathbf{g}_m] \quad (3.22)$$

Among them,

$$g_l = \begin{bmatrix} g_l^{(1,1)} & g_l^{(1,2)} & \cdots & g_l^{(1,n_0)} \\ g_l^{(2,1)} & g_l^{(2,2)} & \cdots & g_l^{(2,n_0)} \\ \vdots & \vdots & \ddots & \vdots \\ g_l^{(k_0,1)} & g_l^{(k_0,2)} & \cdots & g_l^{(k_0,n_0)} \end{bmatrix}$$

Thus, the semi-infinite generation matrix is

$$\mathbf{G}_\infty = \begin{bmatrix} g_\infty & 00 \cdots 0 & 00 \cdots 0 & \cdots \\ 00 \cdots 0 & g_\infty & 00 \cdots 0 & \cdots \\ 00 \cdots 0 & 00 \cdots 0 & g_\infty & 00 \cdots 0 \cdots \\ \cdots & & & \end{bmatrix} = \begin{bmatrix} \mathbf{g}_0 & \mathbf{g}_1 & \cdots & \mathbf{g}_m & \mathbf{O} & \cdots \\ \mathbf{O} & \mathbf{g}_0 & \mathbf{g}_1 & \cdots & \mathbf{g}_m & \mathbf{O} \cdots \\ \mathbf{O} & \mathbf{O} & \mathbf{g}_0 & \mathbf{g}_1 & \cdots & \mathbf{g}_m \cdots \\ \cdots & & & & & \end{bmatrix} \quad (3.23)$$

It can be observed that for the general convolutional code (n_0, k_0, m) , its generation matrix $\mathbf{G}(D)$ is a matrix of order $k_0 \times n_0$, and its elements represent the relationship between the n_0 output codes generated at each moment in the convolutional code and the k_0 input codes. Because the linear convolutional code is considered, after the generation matrix \mathbf{G}_∞ is obtained, the relationship between the generation matrix and check matrix is determined.

$$\mathbf{G}_\infty \mathbf{H}_\infty^T = \mathbf{O} \quad (3.24)$$

The consistency check matrix of the convolutional code was obtained. For the (n_0, k_0, m) convolutional code, the check matrix has the following form:

$$\mathbf{H}_\infty = \begin{bmatrix} \mathbf{h}_0 & \mathbf{O} & \mathbf{O} & \cdots \\ \mathbf{h}_1 & \mathbf{h}_0 & \mathbf{O} & \cdots \\ \mathbf{h}_2 & \mathbf{h}_1 & \mathbf{h}_0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \cdots \\ \mathbf{h}_m & \mathbf{h}_{m-1} & \cdots & \mathbf{h}_0 \cdots \\ \mathbf{O} & \mathbf{h}_m & \cdots & \mathbf{h}_1 \cdots \\ \mathbf{O} & \mathbf{O} & \cdots & \mathbf{O} \cdots \\ \vdots & \vdots & & \vdots \end{bmatrix} \quad (3.25)$$

The elements in the matrix are all $(n_0 - k_0) \times k_0$ -order matrices. From Eq. (3.24), we can calculate all elements of \mathbf{H}_∞ . The \mathbf{H}_∞ matrix is also a semi-infinite matrix. The corresponding check-function matrix is defined as a matrix of the order $(n_0 - k_0) \times k_0$:

$$\mathbf{H}(D) = \begin{bmatrix} h^{(1,1)}(D) & h^{(1,2)}(D) & \cdots & h^{(1,n_0)}(D) \\ h^{(2,1)}(D) & h^{(2,2)}(D) & \cdots & h^{(2,n_0)}(D) \\ \vdots & \vdots & \ddots & \vdots \\ h^{(n_0-k_0,1)}(D) & h^{(n_0-k_0,2)}(D) & \cdots & h^{(n_0-k_0,n_0)}(D) \end{bmatrix} \quad (3.26)$$

Among them,

$h^{(i,j)}(D) = h_0^{(i,j)} + h_1^{(i,j)}D + \cdots + h_m^{(i,j)}D^m$, $i = 1, 2, \dots, n_0 - k_0$, $j = 1, 2, \dots, n_0$ is the generator function of the code with respect to the i th check output. The parity matrix $\mathbf{H}(D)$ of the convolutional codes reflects the relationship between the $n_0 - k_0$ parity elements and the former k_0 information elements in the codeword.

3.3 Turbo-Code Interleavers

The interleaver is a very important component in the turbo-code system. The interleaver is mainly used to reduce the correlation between the check bits, and thus reduce the bit-error rate in the iterative decoding process.

3.3.1 Basic Interleaver Concepts

Interleaving is the process of resetting the positions of elements in the data sequence to obtain an interleaved sequence. The reverse of this process is to restore the original order of the elements in the interleaving sequence, based on the interleaving sequence, thus restoring the original sequence. This is also called the process of deinterlacing. The equipment that interleaves and deinterleaves is called the interleaver and deinterleaver [7].

For example, let the input of interleaver I be

$$u = (u_1, u_2, \dots, u_N),$$

where $u_i \in \{0, 1\}$, $i = 1, 2, \dots, N$.

The output sequence of interleaved mapping is denoted as

$$\tilde{u} = (\tilde{u}_1, \tilde{u}_2, \dots, \tilde{u}_N),$$

where $\tilde{u}_j \in \{0, 1\}$, $j = 1, 2, \dots, N$.

Sequence \tilde{u} differs from sequence u only in the order in which the elements are placed. If the input sequence and interleaving sequence are regarded as a pair of sets containing N elements, the interleaving process can be regarded as a one-to-one mapping process from set u to set \tilde{u} , namely,

$$I : u_i \rightarrow \tilde{u}_j,$$

if the set is defined as

$$S = \{1, 2, \dots, N\}.$$

The interleaving process can also be regarded as a mapping-index function

$$I(A \rightarrow A) : j = I(i), \quad i, j \in A,$$

where i and j are the indices of elements in the original sequence u and the interlaced sequence \tilde{u} , respectively. The mapping function is represented by an interleaving vector:

$$I_N = \{I(1), I(2), \dots, I(N)\}.$$

3.3.2 Effects of the Interleaver on Turbo-Code Performance

The role of the interleaver in a turbo encoder is to reorder the bits in an information sequence. When the weight of the message sequence encoded by the first component encoder is low, the interleaver can improve the weight of the message sequence encoded by the second component encoder so that it outputs high-weight codewords with a high probability, thereby increasing the Hamming weight of the codeword. In addition, a good interleaver can effectively reduce the correlation between check sequences. Through interlacing, the encoding sequence is memoryless, in the range of $2N$ or $3N$ bits (without deletion), and an approximate random long code is constructed from a simple short code.

From the information-theory perspective, the purpose of introducing an interleaver into the turbo-code encoder is to randomize the coding; however, it is impossible to randomize the coding, given the limited interleaver length. For an input information sequence with limited length, it is impossible to achieve complete randomness; therefore, an interleaver based on the randomness criterion is usually called a pseudo-random interleaver. The shorter the interleaver length, the worse the randomness. In this case, an interleaver designed according to certain deterministic rules can achieve better performance than a pseudo-random interleaver. Generally, the performance of the interleaver is measured using the following criteria [6, 8].

The first is the distance between the bits before and after the interlacing. If the interleaver can space out the information bits that are close to each other in the original sequence, the turbo-code performance can be improved to a certain extent.

Second, the performance of turbo codes can be improved if the designed interleaver can evenly protect the system bits. Consider the turbo codes in Fig. 3.1 as an example. Turbo codes with a bit rate of $1/3$ were deleted to obtain a turbo code with

a bit rate of $1/2$. In the output code, the numbers of information bits and parity bits are the same.

Suppose that the deletion process is implemented by deleting the odd bits in the check sequence output by the first component code and the even bits in the check sequence output by the second component code. When using a pseudo-random interleaver, a bit in an odd position (i.e., the information subscript is odd) in the original sequence may appear at an even position after the interleaver.

To implement turbo-code deletion, the two parity bits corresponding to certain information bits are deleted or preserved. Only one parity bit for the other information bits is preserved. The probability of errors in information bits where both check bits are deleted during decoding is greatly increased, thus affecting the turbo-code performance. Therefore, if the designed interleaver can maintain the parity of the information-bit position, it will help to improve the performance.

Third, the turbo-code encoder and decoder must have a corresponding deinterleaver in addition to the interleaver, to reverse the process of interleaving. Therefore, if the designed interleaver satisfies the symmetric characteristics, the interleaver and deinterleaver are exactly the same and can be realized using the same device. This reduces the complexity of the device and further improves the performance of the turbo codes.

3.3.3 Typical Interleavers

The basic ideas of some typical interleavers and pseudo-random interleavers are introduced below, and their corresponding input–output positions are provided for direct comparison.

(1) Block or grouping interleaver

The grouping interleaver is the simplest type. Its interleaving mapping process can be described as follows. The data sequence is written into the matrix in row order and then read out in column order; thus, the interleaving is completed. In the corresponding deinterleaving process, the interleaved data sequence is written in column order and then read in row order. Figure 3.3 shows the interleaving process [9].

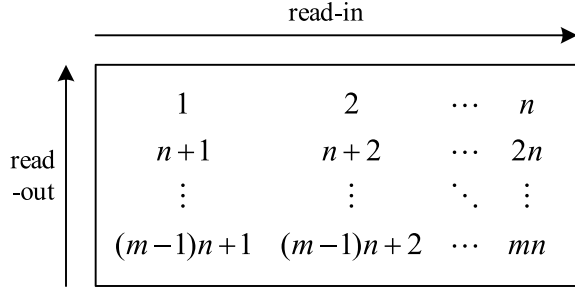
The interleaving mapping function of the grouping interleaver can be expressed as

$$I(i) = [(i - 1) \bmod n] + [(i - 1)/n] + 1, \quad i = 1, 2, \dots, N, \quad (3.27)$$

where N is the interlace length.

Block interleaving can ensure that the adjacent bits in the original sequence have a certain distance after interleaving, which is helpful for improving the performance of turbo codes. In practice, grouping interlaced with shorter interlaced lengths can result in better performance.

Fig. 3.3 Schematic diagram of grouping-interleaving mapping



(2) Packet-spiral interleaver

The block-helical interleaver first writes the data sequence in row order to an $m \times n$ matrix, where m and n are coprime. When interleaving, the data are read from the upper left corner of the matrix to the lower right, and the data move one bit to the right for each lower row (i.e., the column index increases as the row index increases, and the increment step is 1). Modulo m and n are taken for the indexes in the row and column directions, respectively; that is, if r_i and c_i represent the row index and column index of the i th bit, respectively, the data-reading sequence of the grouping-spiral interleaver is

$$\begin{aligned} r_{i+1} &\equiv r_i + 1 \pmod{m} \\ c_{i+1} &\equiv c_i + 1 \pmod{n}, \end{aligned} \quad (3.28)$$

where $i = 0, 1, \dots, N - 1$.

The initial values of the above recursive operation are

$$\begin{aligned} r_0 &= 0 \\ c_0 &= 0 \end{aligned} \quad (3.29)$$

Figure 3.4a illustrates the interleaving process of a grouping-spiral interleaver [6]. In addition, the grouping-spiral interleaving process can also read data from the lower left corner of the interleaving matrix; the corresponding data-reading sequence is

$$\begin{aligned} r_{i+1} &\equiv r_i - 1 \pmod{m} \\ c_{i+1} &\equiv c_i + 1 \pmod{n}, \end{aligned} \quad (3.30)$$

where $i = 0, 1, \dots, N - 1$.

The initial values of the recursive operation are

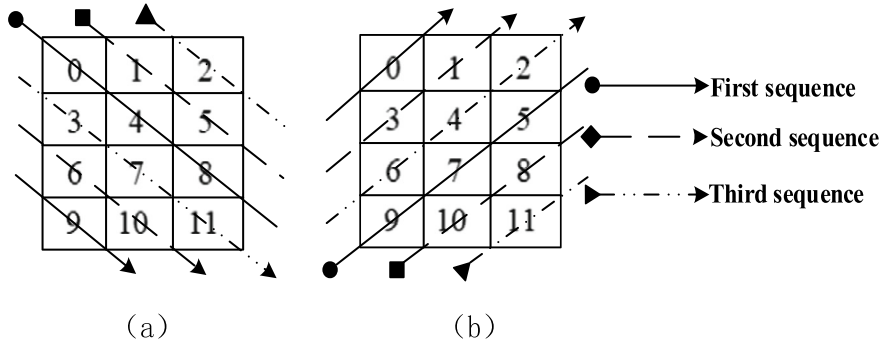


Fig. 3.4 Schematic diagram of grouping-spiral interleaving

$$r_0 = N - 1$$

$$c_0 = 0. \quad (3.31)$$

Figure 3.4b shows the interleaving process

Generally, the design requirements of a grouping-spiral interleaver ($m \times n$) are as follows [6]:

1. $m + 1$ and n are coprime.
2. n should be a multiple of the encoding-constraint length. When the feedback polynomial of the component codeword is the original polynomial, it can ensure that the ending ratio of the two component encoders returns to zero.
3. In addition, n being even can make the interleaver satisfy the modulo-2 attribute [10].

(3) Pseudo-random interleaver

A pseudo-random interleaver refers to an interleaver generated randomly by interleaver mapping. Each pseudo-random interleaver of length N has $N!$ possible interleavers. The design process of a pseudo-random interleaver of length N is as follows [11].

1. An integer i_1 is randomly selected from the set $S = \{1, 2, \dots, N\}$, and the concept $p(i_1) = 1/N$ of i_1 is selected. The selected i_1 is denoted as $I(1)$; then, i_1 is deleted from set S , and the new set obtained is denoted as S_1 .
2. At step k , an integer i_k is randomly selected from the set $S_{k-1} = \{i \in S, i \neq i_1, i_2, \dots, i_{N-k+1}\}$, and the corresponding selection probability is $p(i_k) = 1/(N - k + 1)$. The selected i_k is denoted as $I(k)$, i_k is deleted from set S_{k-1} , and the new obtained set is denoted as S_k .
3. When $k = N$, $I(N)$ is obtained, and the corresponding selection probability is $p(i_N) = 1$ and $S_N = \varphi$. Finally, the interweaving process ends.

The implementation of a pseudo-random interleaver is described as follows.

From the information-theory perspective, the purpose of introducing an interleaver into a turbo-code encoder is to randomize the coding. Using a random interleaver increases the randomness of the interleaver sequence, and is more consistent with the principle of random coding. Simulation results prove that, in the case of long frames, random interleavers have obvious advantages; unfortunately, they also have many defects [6, 12]:

1. The random interleaver has the best average performance; however, for each interleaver, owing to its randomness, it is impossible to guarantee that the output codeword of this interleaver has the best range of spectrum characteristics.
2. The random interleaver has no definite analytic formula, which greatly complicates the accurate analysis of the system's bit-error performance;
3. The generation and storage of random sequences increase the hardware load of the encoder.

3.3.4 Interleaver Properties

The interleaver attributes related to turbo-code design and performance are introduced here. The main attributes include S -distance, modular k , and symmetry [3, 6, 9, 12].

(1) S -distance property

The extended property or S -distance property of the interleaver implies that the distance between S adjacent bits after interleaving is at least S during interleaving. For pseudo-random interleavers, the S -distance limit should be satisfied when random integers between 1 and N are generated. Such interleavers are known as S -random interleavers. Typically, S -random interleavers are described as follows [6, 12]:

Any

$$|I(i_1) - I(i_2)| \leq S_1, \quad i_1, i_2 \in S \quad (3.32)$$

must meet

$$|i_1 - i_2| \geq S, \quad (3.33)$$

where $I(i_1)$ and $I(i_2)$ are element positions in the original sequence. The limitation of the S -random interleaver is that, when the distance between two elements in the original sequence is less than a certain value S_1 , the distance between the two elements after interleaving must be greater than a given value S . When the two component codes that constitute the turbo codes are identical, $S_1 = S$.

For turbo codes that use convolutional codes as component codes, the probability of burst errors with short lengths is relatively high; therefore, the performance of an S -random interleaver is better than that of a pseudo-random interleaver. The S -random

interleaver can diffuse the adjacent bits of the error sequence after interleaving, and the distance is at least S ; thus, the error is transformed into a random error, which is conducive to the error correction of convolutional codes.

Generally, the performance of turbo codes improves with an increase in the S value. However, the larger the S value is, the more difficult it is to meet the condition, and the more difficult it is to design the interleaver. In general, the following parameter of a better S -random interleaver must be satisfied:

$$S \leq \left\lfloor \sqrt{N/2} \right\rfloor - 1 \quad (3.34)$$

where N is the interlace length.

(2) Modulus k attributes

In turbo codes, to improve the coding efficiency, the bit rates of turbo codes can be increased to $1/2$ by deleting the bits without changing the encoding and decoding structure. However, in general, deleting check bits after interleaving usually results in unequal error protection of the information bits. This is because, after interleaving and deleting, some information bits have two corresponding check bits, whereas others have no corresponding check bits.

The unequal protection of information bits can be eliminated by a proper design of the interleaver. Figure 3.5 illustrates the use of a deletion matrix

$$P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

When a turbo code with a bit rate of $1/2$ is obtained, the two component codes delete the check bits.

In Fig. 3.5, $p(\cdot)$ represents the input sequence u and the check bits corresponding to the interleaved sequences $I(u)$ and $I_2(u)$, where $I(\cdot)$ is the random interleaver and $I_2(\cdot)$ is the interleaver satisfying the modulo-2 properties. It can be seen from Fig. 3.5 that the deletion of the parity bits after the random interleaver is not uniform. In other words, some information bits have two corresponding check bits preserved, some information bits have only one corresponding check bit preserved because one is deleted, and some information bits have both corresponding check bits deleted.

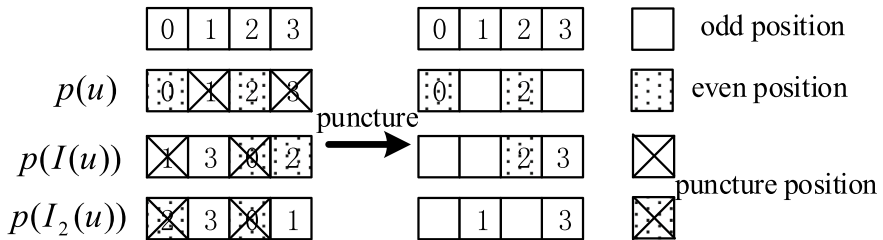


Fig. 3.5 Protection of system bits by the pseudo-random interleaver and modulo-2 interleaver [6]

However, interleaver $I_2(u)$, satisfying the modulo-2 property, can ensure that each information bit has only one check bit to ensure the uniform protection of the information bits.

The concept of a modulo-2 interleaver was first used in the parity interleaver proposed by Barbulescu. The simulation results show that the parity interleaver can improve the performance of the deleted turbo code with a bit rate of 1/2. A modulo-2 intertwined odd location ensures that the information in a sequence of bits is still in the odd position after mixing and the location of a bit after weaving is still in the even position, to ensure that the first component of the encoder outputs an odd parity and the second component of the encoder outputs a parity checking after deleting more corresponding to each information than is characteristic and is only a parity bit.

If the restriction is relaxed to modulo k , $k > 2$,

$$J \bmod k = 0, \quad \text{only when} \quad (3.35)$$

$$I \bmod k = 0 \quad \text{is satisfied.} \quad (3.36)$$

$$\text{If } I \bmod k \in \{1, 2, \dots, k-1\} \quad (3.37)$$

the element that satisfies the position interweaving of

$$J \bmod k \in \{1, 2, \dots, k-1\} \quad (3.38)$$

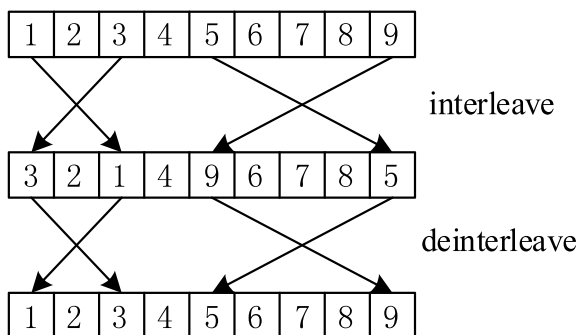
is acceptable. Such interleaving can introduce more randomness while maintaining the properties of redundant code modulo k .

(3) Symmetry properties

The common shortcoming of these interleavers is that they not only need to design interleaver sequences but also need to consider the corresponding deinterleaver sequences in the decoder. Because interleavers and deinterleavers are generally different, different hardware devices and lookup tables are required for the two sequences. This problem can be solved by selecting a symmetric interleaver. For symmetric interleavers, the interleaving and deinterleaving sequences are exactly the same; therefore, only one interleaving sequence can conduct both the interleaving and deinterleaving processes.

The basic idea of a symmetric interleaver is to exchange the positions of two bits in a sequence. When a sequence is interwoven twice, the original sequence is obtained. Figure 3.6 illustrates the symmetrical-interleaving process with a symmetrical interleaver of length nine. As can be seen from Fig. 3.6, the interleaving and deinterleaving sequences of the symmetric interleavers are exactly the same, and the interleaving and deinterleaving processes are consistent. In other words, the mapping process for symmetric interleaving is $I \rightarrow I$.

Fig. 3.6 Schematic diagram of symmetrical interleaving



(4) Double-return-to-zero property

In turbo-code encoding, for the first component encoder, the end bit corresponding to the encoding register state can be added to the end of the data sequence to make it zero. However, because the interleaver exists, the data sequence input by the second component encoder is the data sequence after the interleaver, and its ending bit generally does not correspond to the encoding state; therefore, the encoding grid will not be zero at the end of encoding.

If the interleaver satisfies certain characteristics by adding certain restrictions, the two encoder grids can be zeroed simultaneously. If the second component of the encoder of the figure is zero, then the bit corresponding to the second component of the decoder's initial state is known, allowing it to decode the recursive calculation after the initial value of the conditional probability is determined. Thus, we can improve the decoding output of external information and the reliability of the output of soft information and, to a certain extent, improve the performance of turbo-code interleavers. The property that can reset the grid of two component encoders simultaneously is generally called the double-return-to-zero, and the corresponding interleaver is called a double-return-to-zero interleaver.

For a given interleaving length N , if the two component codes constituting turbo codes are exactly the same, the necessary and sufficient condition for the interleaver to make the two component codes double-return-to-zero is, for any $i = 0, 1, \dots, N-1$,

$$I(i) \equiv i \pmod{p} \quad (3.39)$$

In other words, if the interleaving mapping relationship satisfies Eq. (3.39), the interleaver interleaves each unit vector in the subset to which it belongs; that is, for any $i = 0, 1, \dots, N-1$, if $e_i \in U_j$, there must be $I(e_i) = e_{I(i)} \in U_j$; thus,

$$f(e_{I(i)}) = f(e_i) \quad (3.40)$$

For example, for an $m \times n$ grouping interleaver, we define the following mapping:

$$i \rightarrow I(i) = (i \bmod n)m + i \operatorname{div} n \quad (3.41)$$

The necessary and sufficient condition for the double-return-to-zero of the coded grid is for any $i = 0, 1, \dots, N - 1$,

$$i \equiv [(i \bmod n)m + i \operatorname{div} n] \bmod p \quad (3.42)$$

If $i = 1$

$$1 \equiv m \bmod p \quad (3.43)$$

If $i = n$

$$n \equiv 1 \bmod p \quad (3.44)$$

Thus, the double-return-to-zero block interleaver should have $m = jp + 1$ rows and $n = lp + 1$ columns, where j and l are positive integers.

For example, when designing the generator for a (7, 5) polynomial turbo code, the component for a period of three-yard feedback polynomials, the size of 13×13 groups intertwined in the mapping relationship must meet Eq. 3.42, which is double-zero intertwined; then, the length of the input sequence is 169, of which the last two bits are at the end. In addition, the double-return-to-zero interleaver satisfies the modulo 2 property.

3.4 Decoding Turbo Codes

One of the fundamental reasons for the excellent performance of turbo codes is the use of iterative decoding, which improves the decoding performance by exchanging soft information between component decoders. The complexity of iterative decoding only increases linearly with an increase in the information-sequence size. Compared with optimal maximum-likelihood decoding (MLD), whose decoding complexity increases exponentially with an increase in the codeword length, iterative decoding has better realizability. To achieve better turbo-code decoding performance, the soft-in soft-out (SISO) algorithm is adopted to decode component codes, to exchange soft information between component decoders during iterative decoding.

3.4.1 Turbo-Code Decoding Structure

Figure 3.7 shows the typical decoding structure of a turbo code [13]. In turbo-code iterative decoding, the symbolic log-likelihood ratio $\Lambda(u; I)$ (prior information) of

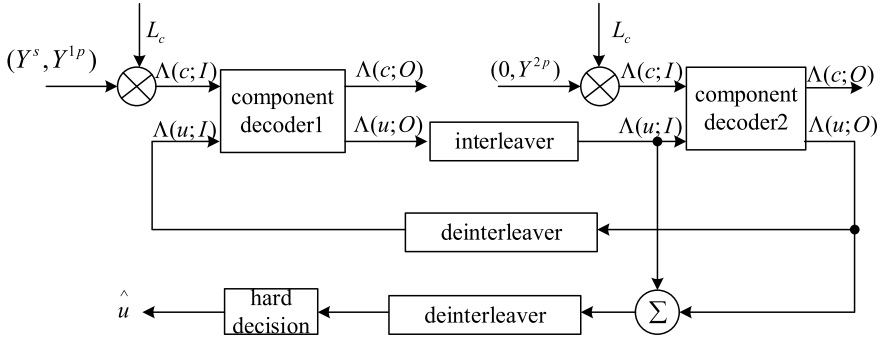


Fig. 3.7 Structure of the turbo-code iterative decoder [13]

the input information is initialized to zero at the first iteration. The symbolic log-likelihood ratio $\Lambda(u; O)$ of the output information after decoding is used as the symbolic log-likelihood ratio of the input information after interleaving:

$$\Lambda^{(2)}(\mathbf{u}; \mathbf{I}) = \Lambda_I^{(1)}(\mathbf{u}; \mathbf{O}) \quad (3.45)$$

The superscript represents different component-decoding modules, and the subscript I represents the interleaving process.

The decoder 2 decoding module decodes the output at the end of the first iteration. Then, it generates the information-symbol probability of the logarithmic-likelihood ratio after mixed feedback to the decoder1 decoding module, as prior information for the next round of decoding (input information-symbol probability logarithmic-likelihood ratio):

$$\Lambda^{(1)}(\mathbf{u}; \mathbf{I}) = \Lambda_{I^{-1}}^{(2)}(\mathbf{u}; \mathbf{O}) \quad (3.46)$$

where I^{-1} stands for deinterleaving. The above process is repeated until a certain number of iterations is reached or certain iteration stop conditions are met. Finally, a hard decision is made according to the symbolic-probability log-likelihood ratio $\Lambda(u; O)$ of the output information of component decoder 2.

3.4.2 Soft-Output Decoding Algorithm Based on a Posterior Probability

An important characteristic of turbo codes is that they use iterative decoding, whose complexity increases linearly. In the iteration process, soft-bit information is exchanged between component decoders to improve the decoding performance.

Forney et al. proved that the optimal soft-output decoder should be an a posteriori probability decoder, which is the transmission probability of a particular bit, conditional on receiving a signal.

(1) MAP decoding algorithm

Nearly 20 years after the maximum a posteriori (MAP) algorithm was proposed, it had not received much attention because of the high computation and hardware-implementation complexity. It was not until 1993, when the inventor of the turbo code adopted it in his original turbo iterative-decoding scheme, that people began to study the algorithm again, and proved that it was the best suboptimal algorithm for turbo iterative decoding.

The MAP algorithm is a soft-output decoding algorithm based on codeword lattices that minimize the bit-error probability. According to the MLD principle, the main task of the decoder is calculated under the condition of receiving the sampling probability of different sent symbols, namely, $p(u_k = u | Y)$. It then receives the sample sentence for the probability value of the largest information symbol [5]:

$$\hat{u} = \arg(\max_{u: u_j \in U} p(u_j = u | Y)) \quad (3.47)$$

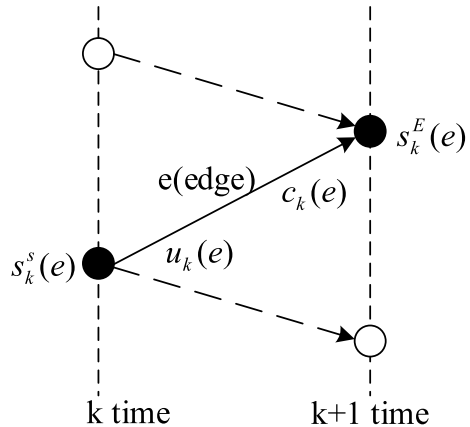
First, we define the concept of “edge” in a coding grid, as shown in Fig. 3.8 [6].

$S_k^S(e)$ and $S_k^E(e)$ are the starting and ending states of edge e respectively. $u_k(e)$ is the information symbol, $c_k(e)$ is the code character number, and k is the time index. It can be seen that at time k , the input bit and start state uniquely determine the output codeword, end state, and edges in the graph.

Following are some symbol definitions:

- u_k : Bit of input information at time k ;
- U : Information sequence set;
- c_k : Coded output codeword at moment k ;
- C : Codeword sequence set;

Fig. 3.8 Concept of edges in a grid



X_k : Symbol sent at time k ;

Y_k : Symbol received at time k ;

X_i^j : Sequence from signal X_i at moment i to X_j at moment j in the transmitted symbol;

Y_i^j : Sequence from signal Y_i at time i to Y_j at time j in the received symbol.

For binary phase-shift keying (BPSK) modulation, the relationship between the sent symbol and the codeword is

$$X_k = \sqrt{E_s}(2c_k - 1), \quad k = 1, 2, \dots, N \quad (3.48)$$

where E_s is the average signal power.

The bit probability $p_k(\mathbf{u}; O)$ is as follows:

$$\begin{aligned} p_k(\mathbf{u}; O) &= p(u_k = u | Y_1^N) \\ &= \frac{1}{p(Y_1^N)} \sum_{u: u(e) \in U} p(s_k^S(e), s_k^E(e), Y_1^{k-1}, Y_k, Y_{k+1}^N) \\ &= \frac{1}{p(Y_1^N)} \sum_{u: u(e) \in U} p(s_k^S(e), Y_1^{k-1}) p(s_k^E(e), Y_k | s_k^S(e)) p(Y_{k+1}^N | s_k^E(e)) \quad (3.49) \\ &= \frac{1}{p(Y_1^N)} \sum_{u: u(e) \in U} \alpha_{k-1}(s_k^S(e)) \gamma_k(e) \beta_k(s_k^E(e)). \end{aligned}$$

Considering that the RSC encoder is equivalent to a Markov source, when state s_k at time k is known, events occurring after time k are unrelated to the previous input. Therefore, probabilities $\alpha_k(s)$ and $\beta_k(s)$ can be obtained by forward and backward recursion, respectively.

Their recursive calculations are as follows:

$$\begin{aligned} \alpha_k(s) &= p(s_k^S(e) = s, Y_1^{k-1}) \\ &= \sum_{e: s_k^E(e)} \alpha_{k-1}(s_k^S(e)) \gamma_k(e), \quad k = 1, 2, \dots, N-1 \quad (3.50) \end{aligned}$$

$$\begin{aligned} \beta_k(s) &= p(Y_{k+1}^N | s_{k+1}^S(e)) \\ &= \sum_{e: s_{k+1}^S(e)} \beta_{k+1}(s_{k+1}^E(e)) \gamma_{k+1}(e), \quad k = N-1, N-2, \dots, 2, 1. \quad (3.51) \end{aligned}$$

Figure 3.9 shows the calculation process.

If the initial state S_0 and end-state S_N of the encoder are known, the initial values of the above recursion can be set as

$$a_0(s) = \begin{cases} 1, & s = S_0 \\ 0, & \text{else} \end{cases} \quad (3.52)$$

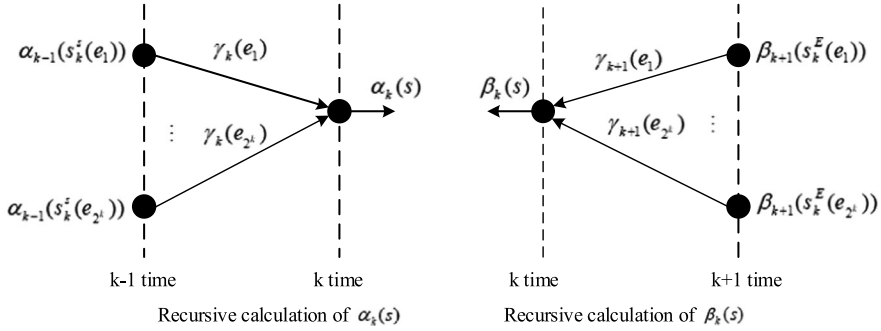


Fig. 3.9 Calculations of $\alpha_k(s)$ and $\beta_k(s)$

$$\beta_N(s) = \begin{cases} 1, & s = S_N \\ 0, & \text{else} \end{cases} \quad (3.53)$$

If the end state of the encoder is unknown, on the premise that the probability of the input bit taking “0” or “1” is equal, the probability of the register obtaining all states at the end of the encoding is the same; therefore, the initial value of the backward recursion can be set as

$$\beta_N(s) = \frac{1}{2^v}, \quad \text{all } s, \quad (3.54)$$

where v is the number of registered encoder units.

To prevent overflow, the following factors can be normalized by $\alpha_k(s)$ and $\beta_k(s)$:

$$L = \frac{1}{\max \alpha_k(s)} \quad (3.55)$$

$$M = \frac{1}{\max \beta_k(s)} \quad (3.56)$$

$\gamma_k(e)$ is the measure of the edge, and using the Bayesian formula, we obtain

$$\gamma_k(e) = p(s_k^E(e), Y_k | s_k^S(e)) = \sum_{u: m_k \in U} p(s_k^E(e) | s_k^S(e)) p(X_k | e) p(Y_k | X_k) \quad (3.57)$$

The first term is the state-transition probability of a branch (edge) in the lattice graph, which is determined by the prior information $\Lambda_a(u_k)$ of the information bits [6, 14, 15]:

$$p(s_k^E(e)|s_k^S(e)) = \begin{cases} p(u_k = 1) = \frac{\exp(\Lambda_a(u_k))}{1 + \exp(\Lambda_a(u_k))}, & u_k = 1 \\ p(u_k = 0) = \frac{1}{1 + \exp(\Lambda_a(u_k))}, & u_k = 0 \end{cases}, \quad (3.58)$$

$$\text{where } \Lambda_a(u_k) = \lg \frac{p(u_k = 1)}{p(u_k = 0)}. \quad (3.59)$$

The second term $p(X_k | e)$ is 1 or 0, depending on whether X_k is related to edge e ; the term varies depending on the channel model. For example, for BPSK-modulation transmissions on additive white Gaussian noise (AWGN) channels,

$$p(Y_k | X_k) = \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{(Y_k - \sqrt{E_s}(2u_k - 1))^2}{N_0}\right). \quad (3.60)$$

For binary inputs, the log-likelihood ratio (LLR) can be used as the decision function:

$$\begin{aligned} \Lambda(u_k) &= \lg \frac{p(u_k = 1 | Y_1^N)}{p(u_k = 0 | Y_1^N)} \\ &= \lg \frac{\sum_{u:u(e)=1} \alpha_{k-1}(s_k^S(e)) \gamma_k(e) \beta_k(s_k^S(e))}{\sum_{u:u(e)=0} \alpha_{k-1}(s_k^S(e)) \gamma_k(e) \beta_k(s_k^S(e))}. \end{aligned} \quad (3.61)$$

The MAP algorithm makes decisions based on the value of $\Lambda(u_k)$:

$$\hat{u}_k = \begin{cases} 1, & \Lambda(u_k) \geq 0 \\ 0, & \Lambda(u_k) \leq 0 \end{cases}. \quad (3.62)$$

Similar to Eq. (3.49), the posterior probability of the output codeword of the MAP algorithm can be obtained as follows:

$$\begin{aligned} p_k(\mathbf{u}; O) &= p(c_k = c | Y_1^N) \\ &= \frac{1}{p(Y_1^N)} \sum_{e:c(e) \in C} \alpha_{k-1}(s_k^S(e)) \gamma_k(e) \beta_k(s_k^E(e)), \end{aligned} \quad (3.63)$$

where $1/p(Y_1^N)$ is a constant for a given sequence of information that can be ignored in the calculation.

(2) Log-MAP decoding algorithm

The log-MAP algorithm is a converted form of the MAP algorithm and is simpler to implement. The log-MAP algorithm is implemented by converting all the variables in the MAP algorithm to logarithmic form, thus converting all the multiplication operations to addition operations. In addition, the input and output of the decoder

are correspondingly modified to the form of a logarithmic-likelihood ratio, and then the log-MAP algorithm is obtained by modifying the algorithm [6, 16].

In the log-MAP algorithm, $M_k(e)$, $A_k(s)$, and $B_k(s)$ correspond to $\gamma_k(e)$, $\alpha_k(s)$, and $\beta_k(s)$ in the MAP algorithm, respectively, and they satisfy a logarithmic relationship:

$$M_k(e) = \ln \gamma_k(e) \quad (3.64)$$

$$A_k(s) = \ln \alpha_k(s) \quad (3.65)$$

$$B_k(s) = \ln \beta_k(s). \quad (3.66)$$

First, the calculation of logarithmic-path metric $M_k(e)$ is analyzed. In the derivation of the MAP algorithm, it is known that $p(X_k | e)$ in the $\gamma_k(e)$ expression is 1 or 0, depending on whether X_k is related to edge e . The value of probability $p(s_k^E(e)|s_k^S(e))$ is determined by the prior information $\Lambda_a(u_k)$ of the information bits, and the logarithm of Eq. (3.58) is modified:

$$\ln p(s_k^E(e)|s_k^S(e)) = \begin{cases} \Lambda_a(u_k) - \ln(1 + \exp(\Lambda_a(u_k))), & u_k = 1 \\ -\ln(1 + \exp(\Lambda_a(u_k))), & u_k = 0 \end{cases}. \quad (3.67)$$

For an AWGN channel with $N_0/2$ noise variance, we have

$$\begin{aligned} p(Y_k|X_k) &= p(y_k^1|u_k)p(y_k^2|u_k, s_k^S(e), s_k^E(e)) \\ &= \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{(y_k^s - x_k^s)^2}{N_0}\right) \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{(y_k^p - x_k^p)^2}{N_0}\right). \end{aligned} \quad (3.68)$$

We ignore the constant term:

$$\ln p(Y_k|X_k) = \frac{2y_k^s x_k^s}{N_0} + \frac{2y_k^p x_k^p}{N_0} = \frac{2\sqrt{E_s}}{N_0} [y_k^s(2c_k^s - 1) + y_k^p(2c_k^p - 1)]. \quad (3.69)$$

For the AWGN channel,

$$\Lambda_k(\mathbf{c}^s; \mathbf{I}) = \ln \frac{p_k(c_k^s = 1; \mathbf{I})}{p_k(c_k^s = 0; \mathbf{I})} = \frac{4\sqrt{E_s}}{N_0} y_k^s \quad (3.70)$$

$$\Lambda_k(\mathbf{c}^p; \mathbf{I}) = \ln \frac{p_k(c_k^p = 1; \mathbf{I})}{p_k(c_k^p = 0; \mathbf{I})} = \frac{4\sqrt{E_s}}{N_0} y_k^p. \quad (3.71)$$

Thus, Eq. (3.69) can be written as

$$\ln p(Y_k|X_k) = \frac{1}{2} \Lambda_k(\mathbf{c}^s; \mathbf{I})(2c_k^s - 1) + \frac{1}{2} \Lambda_k(\mathbf{c}^p; \mathbf{I})(2c_k^p - 1). \quad (3.72)$$

Thus, the calculation formula for the path measure in a logarithmic domain can be obtained:

$$M_k(e) = \ln \gamma_k(e) = \ln p(s_k^E(e)|s_k^S(e)) + \ln p(Y_k|X_k) \\ = \begin{cases} \Lambda_a(u_k) - \ln(1 + \exp(\Lambda_a(u_k))) + \frac{1}{2} \Lambda_k(\mathbf{c}^S; \mathbf{I}) + \frac{1}{2} \Lambda_k(\mathbf{c}^P; \mathbf{I})(2c_k^P - 1), & u_k = 1 \\ -\ln(1 + \exp(\Lambda_a(u_k))) - \frac{1}{2} \Lambda_k(\mathbf{c}^S; \mathbf{I}) + \frac{1}{2} \Lambda_k(\mathbf{c}^P; \mathbf{I})(2c_k^P - 1), & u_k = 0 \end{cases} \quad (3.73)$$

We note that Eq. (3.73) holds, only if there is a transfer between $s_k^E(e)$ and $s_k^S(e)$.

Although the information bit sequence of the second component code is not transmitted during encoding, it is actually interlaced with the system bits of the first component code because the system input $\Lambda_k(\mathbf{c}^S; \mathbf{I})$ to the decoder was obtained by weighting the system bits of the received signal. Therefore, $\Lambda_{2k}(\mathbf{c}^S; \mathbf{I})$ of the second component code can be obtained by interleaving $\Lambda_{1k}(\mathbf{c}^S; \mathbf{I})$ of the first component code. Thus, both the first and second component codes have corresponding $\Lambda_k(\mathbf{c}^S; \mathbf{I})$ and $\Lambda_k(\mathbf{c}^P; \mathbf{I})$ for the information bits. Therefore, the same calculation formula can be used by both the component decoders to calculate the forward-path metric $A_k(s)$ and backward-path metric $B_k(s)$.

Because exponentials and calculations exist in the recursive operations of $\alpha_k(s)$ and $\beta_k(s)$ in the MAP algorithm (introduced by the calculation of $\gamma_k(e)$ on the AWGN channel), the $\max^*(\cdot)$ operation is introduced into the log-MAP algorithm [6, 17]. It is defined as

$$\max_e^*(f(e)) = \ln \left(\sum_e e^{f(e)} \right), \quad (3.74)$$

thus,

$$A_k(s) = \ln \sum_{e: s_k^E(s)=s} \alpha_{k-1}(s_k^E(e)) \gamma_k(e) \\ = \max_{e: s_k^S(e)=s}^* [A_{k-1}(s_k^E(e)) + M_k(e)], \quad k = 1, 2, \dots, N-1 \quad (3.75)$$

$$B_k(s) = \ln \sum_{e: s_k^E(s)=s} \beta_{k+1}(s_{k+1}^E(e)) \gamma_{k+1}(e) \\ = \max_{e: s_k^S(e)=s}^* [B_{k+1}(s_{k+1}^E(e)) + M_{k+1}(e)], \quad k = N-1, N-2, \dots, 1. \quad (3.76)$$

It is usually possible to transform the above $\max^*(\cdot)$ operation for the case of two variables (x and y) as

$$\begin{aligned}\max^*(x, y) &= \ln(e^x + e^y) = \max(x, y) + \ln(1 + e^{-|x-y|}) \\ &= \max(x, y) + f_c(|x - y|).\end{aligned}\quad (3.77)$$

Assuming that the starting and ending states of the encoder are S_0 and S_N , respectively, then, corresponding to the log-MAP algorithm, the initial recursive calculation values of the forward- and backward-path measures are, respectively,

$$A_0(s) = \begin{cases} 0, & s = S_0 \\ -\infty & \text{else} \end{cases} \quad (3.78)$$

$$B_N(s) = \begin{cases} 0, & s = S_N \\ -\infty & \text{else} \end{cases} \quad (3.79)$$

In the actual numerical calculations, a larger value can be substituted for ∞ . If the state of the encoding register is unknown at the end of encoding, the initial value can be set as

$$B_N(s) = 0 \text{ or other constant.} \quad (3.80)$$

According to the above derivation, the complete logarithmic-likelihood ratio output information of the information bits can be obtained:

$$\begin{aligned}\Lambda_k(\mathbf{u}; \mathbf{O}) &= \ln \sum_{e:u(e)=1} \alpha_{k-1}(s_k^S(e)) \gamma_k(e) \beta_k(s_k^E(e)) \\ &\quad - \ln \sum_{e:u(e)=0} \alpha_{k-1}(s_k^S(e)) \gamma_k(e) \beta_k(s_k^E(e)) \\ &= \max_{e:u(e)=1} * [A_{k-1}(s_k^S(e)) + M_k(e) + B_k(s_k^E(e))] \\ &\quad - \max_{e:u(e)=0} * [A_{k-1}(s_k^S(e)) + M_k(e) + B_k(s_k^E(e))].\end{aligned}\quad (3.81)$$

Substituting the expression of $M_k(e)$ (Eq. 3.73) into Eq. (3.81), and extracting the general term, we obtain

$$\begin{aligned}
\Lambda_k(\mathbf{u}; \mathbf{O}) &= \max_{e:u(e)=1} *[\Lambda_a(u_k) - \ln(1 + \exp(\Lambda_a(u_k))) + \frac{1}{2}\Lambda_k(\mathbf{c}^s; \mathbf{I}) \\
&\quad + \frac{1}{2}\Lambda_k(\mathbf{c}^p; \mathbf{I})(2c_k^p - 1) + A_{k-1}(s_k^E(e))] \\
&\quad - \max_{e:u(e)=0} *[-\ln(1 + \exp(\Lambda_a(u_k))) - \frac{1}{2}\Lambda_k(\mathbf{c}^s; \mathbf{I}) \\
&\quad + \frac{1}{2}\Lambda_k(\mathbf{c}^p; \mathbf{I})(2c_k^p - 1) + A_{k-1}(s_k^E(e))] \\
&= \left(\Lambda_a(u_k) - \ln(1 + \exp(\Lambda_a(u_k))) + \frac{1}{2}\Lambda_k(\mathbf{c}^s; \mathbf{I}) \right) \\
&\quad + \max_{e:u(e)=1} * \left[A_{k-1}(s_k^E(e)) + \frac{1}{2}\Lambda_k(\mathbf{c}^p; \mathbf{I})(2c_k^p - 1) + B_k(s_k^E(e)) \right] \\
&\quad - \max_{e:u(e)=0} * \left[A_{k-1}(s_k^E(e)) + \frac{1}{2}\Lambda_k(\mathbf{c}^p; \mathbf{I})(2c_k^p - 1) + B_k(s_k^E(e)) \right] \\
&= \Lambda_a(u_k) + \Lambda_k(\mathbf{c}^s; \mathbf{I}) \\
&\quad + \max_{e:u(e)=1} * \left[A_{k-1}(s_k^E(e)) + \frac{1}{2}\Lambda_k(\mathbf{c}^p; \mathbf{I})(2c_k^p - 1) + B_k(s_k^E(e)) \right] \\
&\quad - \max_{e:u(e)=0} * \left[A_{k-1}(s_k^E(e)) + \frac{1}{2}\Lambda_k(\mathbf{c}^p; \mathbf{I})(2c_k^p - 1) + B_k(s_k^E(e)) \right].
\end{aligned} \tag{3.82}$$

It can be seen from Eq. (3.82) that the output logarithmic-likelihood ratio information $\Lambda_k(u; \mathbf{O})$ is the sum of prior information $\Lambda_a(u_k)$, system information $\Lambda_k(\mathbf{c}^s; \mathbf{I})$, and external information (the remaining part).

As the two component decoders use the same system information $\Lambda_k(\mathbf{c}^s; \mathbf{I})$, it must be separated from $\Lambda_k(u; \mathbf{O})$ because prior information $\Lambda_a(u_k)$ is processed, and only external information is used as prior information for the next round of decoding. The external information is as follows:

$$\begin{aligned}
\Lambda_e(u_k) &= \Lambda_k(\mathbf{u}; \mathbf{O}) - \Lambda_a(u_k) - \Lambda_k(\mathbf{c}^s; \mathbf{I}) \\
&= \max_{e:u(e)=1} * \left[A_{k-1}(s_k^E(e)) + \frac{1}{2}\Lambda_k(\mathbf{c}^p; \mathbf{I})(2c_k^p - 1) + B_k(s_k^E(e)) \right] \\
&\quad - \max_{e:u(e)=0} * \left[A_{k-1}(s_k^E(e)) + \frac{1}{2}\Lambda_k(\mathbf{c}^p; \mathbf{I})(2c_k^p - 1) + B_k(s_k^E(e)) \right].
\end{aligned} \tag{3.83}$$

Similarly, the log-likelihood ratio of the decoded output probabilities for code character numbers can be obtained. $\max^*(\cdot)$ in the log-MAP algorithm is simplified to the usual maximum operation; namely, the max-log-MAP algorithm [18].

(3) Soft-output Viterbi decoding algorithm

For convolutional codes, the Viterbi algorithm is the optimal maximum-likelihood decoding method, and the decoding output is the optimal estimation sequence of the

convolutional codes. However, the traditional Viterbi algorithm has two defects for turbo codes, which belong with cascade-convolutional codes.

First, the burst error in the output of one component decoder affects the decoding performance of the other component decoder, thereby degrading the performance of the cascade codes. Second, regardless of whether a soft-decision or hard-decision Viterbi algorithm is used, its decoding output is hard-decision information. If one component code is decoded by the Viterbi algorithm, the other component decoder will only receive hard-decision results as input and cannot perform soft-decision decoding; thus, the performance will be reduced.

If the Viterbi decoder can provide soft-information output, the above two defects can be remedied, and the performance of cascade codes can be significantly improved by exchanging soft information between the component decoders. Therefore, the traditional Viterbi algorithm must be modified to output soft information. The corresponding algorithm is called the soft-output Viterbi algorithm (SOVA) [5, 6].

The decoding process of the SOVA decoder is as follows [6, 19, 20]:

1. Initialization time $t = 0$.

For the zero states in the grid, initialization measure $M_0^{(m)} = 0$. Initialize to ∞ for the other states.

2. $t \leftarrow t + 1$.

Measures are computed for each state in the grid:

$$M_t^{(m)} = M_{t-1}^{(m)} + u_t^{(m)} L_c y_{t,1} + \sum_{j=2}^N x_{t,j}^{(m)} L_c y_{t,j} + u_t^{(m)} \Lambda(u_t),$$

where

- m is the allowable branch/transition to a state in a binary lattice graph, $\mathbf{m} = 1, 2$;
 - $M_0^{(m)}$ is the cumulative measure of m branch paths at time t in the grid.
 - $u_t^{(m)}$ is the input system bit that corresponds to the m branch path at time t in the grid.
 - $x_{t,j}^{(m)}$ is the j^{th} bit in the codeword corresponding to the m branch path at time t in the grid.
 - $y_{t,j}$ is the receiving-channel value corresponding to $x_{t,j}^{(m)}$.
 - L_c is the channel confidence value; $L_c = 4E_b/N_0$, under AWGN-channel conditions.
 - $\Lambda(u_t)$ is the prior information at time t , which is provided by another component decoder.
3. Search $\max_m M_t^{(m)}$ for each state. To simplify the calculation, let $M_t^{(1)}$ represent the measure of the surviving path and $M_t^{(2)}$ represent the measure of the competing path.
 4. Store $M_t^{(1)}$ and the corresponding surviving bits and state paths.

5. Calculate $\Delta_t^0 = \frac{1}{2} |M_t^{(1)} - M_t^{(2)}|$.
6. The surviving and competing paths at each state at time t are compared, and bits on the two paths are stored to determine different relative time values MEM.
7. Update all MEM corresponding measures, $\Delta_t^{MEM} = \min_{k=0,1,\dots,MEM} \{\Delta_t^k\}$, from small to large.
8. Repeat from step 2 until the entire transmission sequence is received.
9. Output the estimated bit sequence \hat{u} and the corresponding soft-output value $\Lambda(\hat{u}) = \hat{u} \cdot \Delta$, where Δ is the last-updated reliability sequence, which is used as prior information for the next round of decoding.

(4). Complexity comparison of several decoding algorithms

This study compares the computational complexity of the MAP, log-MAP, max-log-MAP, and SOVA algorithms [21].

First, the difference between the MAP algorithm and Viterbi algorithm is analyzed. The main difference between the two algorithms is the selection of the decision path in the grid graph. For the soft-decision output at time t , all paths to the final state can be divided into two categories: path set U_0 of $u_k = 0$ and path set U_1 of $u_k = 1$.

The MAP and log-MAP algorithms calculate the sum of the posterior probabilities of all paths in U_0 and U_1 and use the ratio of the two as the soft-output decision of u_k . Therefore, in theory, they should have similar and better decoding performances.

For the SOVA and max-log-MAP algorithms, it can be proven that their hard-decision outputs are the same. They have the same branching-measure function and only a difference in expression. Therefore, the maximum measurement path and corresponding hard-decision output should be the same.

The source of the performance difference between them lies in the different soft-decision-information acquisition methods. Without loss of generality, it is assumed that the maximum path belongs to the all-zero path of U_0 , and the max-log-MAP algorithm selects a path with the maximum metric value from U_i ($i = 0, 1$) and then outputs the difference between the path and the all-zero-path metric value as the soft decision at time k . In other words, the max-log-MAP algorithm selects the path with the maximum magnitudes of U_0 and U_1 instead of summing all path measurements in U_0 and U_1 as does the log-MAP algorithm.

As the signal-to-noise ratio (SNR) increases, one path in both U_0 and U_1 has a much larger metric than the other. At this point, the approximation used in the max-log-MAP algorithm becomes accurate. The SOVA algorithm only backtracks along the maximum metric path (all-zero path, here) to find the path with the maximum metric value that intersects the state node on the all-zero path in path set U_i ($i = 0, 1$), and considers it as the path with the maximum metric value in path set U_i ($i = 0, 1$), which is then used for a soft-decision output. In fact, the path with the largest U_i ($i = 0, 1$) medium magnitude does not necessarily intersect the all-zero path, even if the SNR increases. This leads to errors in the soft-decision information and ultimately affects the decoding performance.

Table 3.1 compares the complexities of several algorithms [6].

Table 3.1 Complexity comparison of different decoding algorithms

Operation	MAP	Log-MAP	Max-log-MAP	SOVA
For maximum		$5 \times 2^v - 2$	$5 \times 2^v - 2$	$2^v + 3(v + 1)$
Addition	4×2^v	$15 \times 2^v + 9$	$10 \times 2^v + 11$	$2 \times 2^v + 8$
Multiplication (division)	$6 \times 2^v + 1$	8	8	8
Lookup		$5 \times 2^v - 2$	0	0

As shown in Table 3.1, the MAP algorithm is the most complex of the four algorithms. The log-MAP algorithm significantly reduces the implementation complexity of the algorithm because it transforms a large number of multiplication operations into addition operations. The max-log-MAP algorithm further reduces the complexity, but also the performance of the algorithm.

The SOVA algorithm is essentially the same as the max-log-MAP algorithm. When $v = 2$, the calculation amount of the SOVA algorithm is approximately twice that of the max-log-MAP algorithm, whereas when $v = 4$, the calculation amount of the max-log-MAP algorithm is almost twice that of the SOVA algorithm.

References

1. Berrou C, Glavieux A, Thitimajshima P (1993) Near Shannon limit error-correcting coding and decoding: turbo-codes(1)[J]. Proc ICC93 2:1064–1070

2. Wolf JK (1978) Efficient maximum likelihood decoding of linear block codes using a trellis[J]. Inf Theory IEEE Transactions 24(1):76–80

3. Donghua L (2002) Research on key techniques of turbo code and the application of turbo principle[D]. National University of Defense Technology, Changsha

4. Trans. Wang Y, Trans. Wang X (1986) Basis and application of error control coding[M]. Posts & Telecom Press, Beijing

5. Wang X, Xiao GZ (2002) Error correcting code, principle and method Revision[M]. Xidian University, Xian

6. Donghua L (2004) Principle and application technology of Turbo code[M]. Publishing House of Electronics Industry, Beijing

7. Andrews K, Heegard C, Kozen D (1997) A theory of interleavers[J]. Cornell University 2:1–10

8. Barbulescu S, Pietrobon SS (1994) Interleaver design for turbo codes[J]. Electron Lett 30(25):2107–2108

9. Carbulescu TH, Wisely DR (1998) Opportunities and challenges for optical wireless: the competitive advantage of free space telecommunications links in today’s crowded market-place[J]. Wirel Technol Syst: Millim-Wave OpticalProc, SPIE 3232:119–128

10. Donghua L, Chaojing T (2000) Design of interleaver in turbo code system[J]. Wirel Commun Technol 1:41–45

11. Donghua L, Chaojing T, Quan Z et al (2000) Improvement of the design of pseudo random Interleaver[J]. Comput & Netw 1:23–24

12. Xu H, Mongtian R, Hongwen Z (2002) Fast interleaver designs for turbo codes[J]. Commun Technol 2:12–14

13. Berrou C, Glavieux A (1996) Near optimum error-correcting coding and decoding: turbo-codes[J]. IEEE Trans Commun 44(10):1261–1271

14. Pierobon SS (1998) Implementation and performance of a turbo/MAP decoder[J]. *Int J Satell Commun* 16(1):23–46
15. Xue-dong WANG, Hua YANG (2002) Implementation of turbo/MAP decoder hardware[J]. *J Harbin Inst Technol* 4:173–176
16. Shujun L, Junhua D, Jianhong Z (2003) Log-MAP algorithm and its implementation of turbo decoder[J]. *Communcations Technol* 2:10–12
17. Jiang J, Bai C, Zhang P, et al (2002) FPGA implementation of turbo decoder for WCDMA system: high efficient structure for Log-MAP turbo decoder[J]. *J Beijing Univ Posts Telecommun* 3:22–26
18. Yuan Y (2005) The study and analysis of MAP algorithm in turbo decoding[D]. Dalian University of Technology, Dalian
19. Zhang L, Wan L, Kuang J (2002) A new SOVA based decoding scheme for Turbo codes[J]. *J China Inst Communcations* 8:24–32
20. Hagenauer J, Hoehner PA (1989) A viterbi algorithm with soft-decision outputs and its applications[C]. In: *Global telecommunications conference*. pp 1680–1686
21. Wang S (2003) Comparison and analysis of the decoding algorithm of turbo-codes and its application in CPM[D]. Nanjing University of Aeronautics and Astronautics, Nanjing

Chapter 4

Error Control Based on LDPC Codes



Low-density parity-check (LDPC) codes are linear error-correction codes based on a sparse matrix and were proposed by Gallager in 1962. Their performance can be very close to the Shannon limit [1–10]. This chapter introduces the encoding and decoding algorithms of LDPC codes as well as their application to channel-coding technology in atmospheric laser communication. Then, a type of semi-random π -rotating LDPC code that can easily be realized in hardware is given. The performance of different typical weather conditions is analyzed using the Simulink simulation tool with pulse-position modulation (PPM).

4.1 LDPC Code Overview

4.1.1 Description of LDPC Codes

LDPC codes are linear block codes defined by the null space of a sparse parity-check matrix H . “Sparse” means that the number of zeros in matrix H is much larger than the number of ones, while “low density” means that the amount of ones in matrix H is very low [3]. If the code length is n and the information bits are k , the check bits are $m = n - k$ and the check matrix H is a matrix of order $m \times n$.

Each row of the checksum matrix represents a checksum constraint, in which the corresponding symbol variables of all non-zero elements form a checksum set, which is represented by a checksum equation. Each column of the checksum matrix represents a checksum constraint in which the symbol participates. The characteristics of a binary (n, j, k) LDPC-code check matrix H are summarized as follows [5–7]:

1. Each column contains j ones; that is, the column weight is j ;
2. Each row contains k ones; that is, the row weight is k ;

1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1

Fig. 4.1 Low-density check matrix [11]. $n = 20, j = 3, k = 4$

3. The number of rows with the same value of 1 between any two columns (called the overlap number) does not exceed 1; that is, in H , there are no four-line cycles in the matrix or Tanner graph, described later;
4. k and k are much less than the code length n and the number of matrix lines m ; when $n \rightarrow \infty, k/n = j/m \rightarrow 0$.

Gallager provides an example based on the above characteristics, as shown in Fig. 4.1.

In addition to the traditional matrix, check matrix H can also be described using the corresponding Tanner bipartite graph [10] (also known as the bipartite graph or factor graph), as follows: The information nodes x_1, x_2, \dots, x_n are arranged in a row corresponding to each column of the check matrix. The information nodes are also called the variable nodes. m check nodes z_1, z_2, \dots, z_m are arranged in a row, and each node corresponds to a check set of the codeword. If the element corresponding to row i and column j of the check matrix is not 0, then node x_j and node z_i are associated and the two nodes are connected; they are referred to as adjacent nodes. The number of edges connected to a node is its degree. Figure 4.2 shows the Tanner diagram of the matrix in Fig. 4.1.

In general, the check matrix is constructed randomly, so it is not systematic. Gaussian elimination can be conducted on the checksum matrix during coding:

$$H = [I \ P], \quad (4.1)$$

where I is the identity matrix and P is an $m \times (n - m)$ -order matrix. The generation matrix can be obtained from Eq. (4.1):

$$G = [-P^T \ I]. \quad (4.2)$$

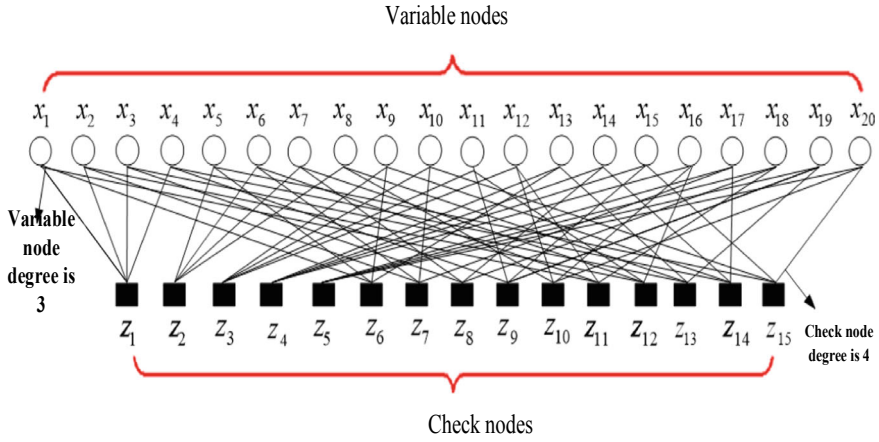


Fig. 4.2 (20, 3, 4) Tanner diagram of an LDPC code [11]

Suppose information sequence $u = (u_0, u_1, \dots, u_{k-1})$; then, the codeword C is

$$C = u \cdot G. \quad (4.3)$$

The LDPC-code coding algorithm consists of three parts: a sparse check matrix, a generation matrix, and codeword generation.

4.1.2 LDPC-Code Loops

Because LDPC codes adopt iterative decoding, the structure of the check matrix has a decisive influence on the performance of the code. The derivation of its algorithm is based on the statistical independence of the information transmitted between nodes. When there is a ring in the bidirectional graph corresponding to matrix H , the inner product of two rows in matrix H is >1 . The information sent by a node will be transmitted back to itself after a loop length, resulting in the superposition of its own information, which destroys the independence, and affects the decoding accuracy.

In bidirectional diagrams, more large rings and fewer small rings are desired; the four shortest rings are especially to be avoided. The appearance of a short ring makes two information nodes participate in two checksums simultaneously. In a sparse checksum matrix, it is impossible to determine which information bit is wrong, if these two checksums fail simultaneously in decoding.

Let any loop length of an LDPC code be L , where $L \geq 4$, and L is a multiple of 2. Messages transmitted during decoding only satisfy the independence hypothesis in the first $L/2$ iterations. To study the influence of loops on the decoding performance, the concept of perimeter length is proposed. The girth refers to the minimum length

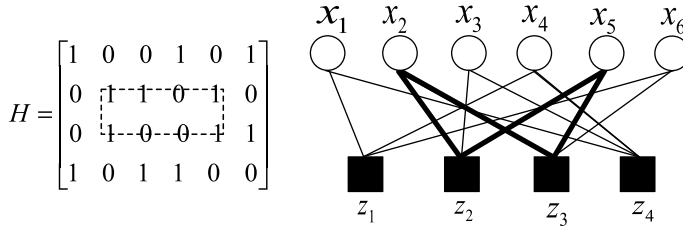


Fig. 4.3 Representation of a ring of four lines in the checksum matrix and bidirectional diagram

of all rings in the Tanner graphs corresponding to an LDPC code. For ring-detection and ring-elimination methods, please refer to the relevant literature [5, 12, 13].

Figure 4.3 shows the corresponding relationship between the LDPC code verification matrix and a ring of four in the bidirectional diagram. The thick line in the bidirectional diagram shows the ring, which corresponds to the elements indicated by the box in the checksum H matrix. Another ring of length four is $x_1 \rightarrow z_1 \rightarrow x_4 \rightarrow z_4 \rightarrow x_1$.

Figure 4.4a shows the characteristic diagram of an LDPC code with rule (3, 6) and code length $N = 300$ without loop cancellation and with short-loop cancellation. It can be seen that when the signal-to-noise ratio (SNR) is small, the characteristic changes before and after the loop cancellation are small; however, when the SNR increases, the characteristic changes before and after the loop cancellation are large. When the bit-error rate is 10^{-7} , the performance of the code after eliminating four rings is about 3 dB better than that with the rings, and the performance after eliminating six rings is about 1 dB better than that after eliminating four rings, with the increase of SNR.

As shown in Fig. 4.4b, when the code length $N = 1000$, when the SNR is < 3 dB, the characteristics of the code before and after the loop is removed are similar. When the SNR is > 3 dB, the performance of the code with the loop does not improve much with the increase of the SNR, showing the flat-layer phenomenon; hence, the occurrence of short loops should be avoided [13].

4.1.3 Classification of LDPC Codes

If the bidirectional graph corresponding to an LDPC code is a regular bidirectional graph, the LDPC code is called a regular LDPC code. If the corresponding bidirectional graph is not a regular bidirectional graph, the LDPC code is called an irregular LDPC code. According to the value of each symbol, LDPC codes can be divided into binary and q -meta LDPC codes. The results show that q -meta LDPC codes are better than binary LDPC codes, and irregular LDPC codes are better than regular LDPC codes.

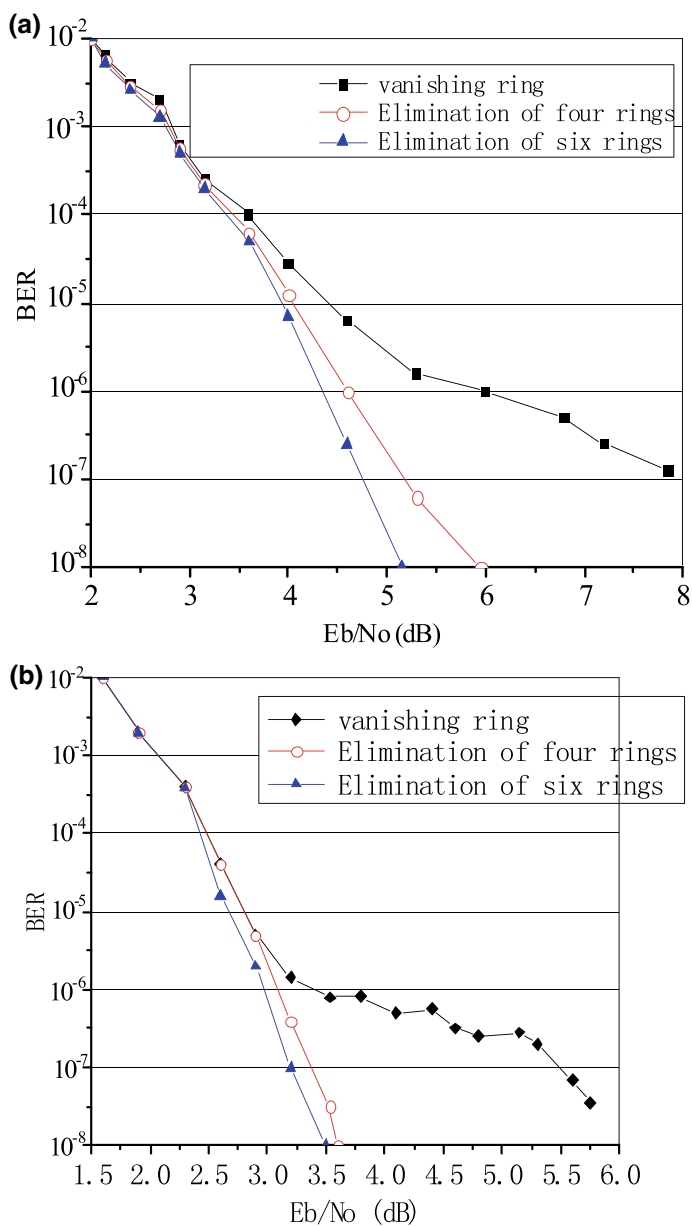


Fig. 4.4 Performance comparison before and after loop elimination with different code lengths [14]. **a** $N = 300$; **b** $N = 1000$

(1) Binary and q -meta LDPC codes

An LDPC code on the finite field $GF(2)$ can naturally be extended to $GF(q)$ ($q = 2^p$, p is an integer), except that the non-zero element of the check matrix H of the LDPC code on $GF(q)$ takes $q - 1$ values, not simply “1”. In this case, each node in the Tanner diagram corresponding to the LDPC code represents a group of codeword bits (parity bits), rather than a single bit (parity bit). LDPC codes in the $GF(q)$ and $GF(2)$ domains have similar decoding ideas.

Research shows that LDPC codes constructed based on $GF(8)$ and $GF(4)$ in additive white Gaussian noise (AWGN) channels have 0.35 and 0.22 dB encoding gains, respectively, when the code length is 18,000 bits, the bit rate is equal to 1/3, and the bit-error rate is 10^{-4} , compared with LDPC codes constructed based on $GF(2)$ under the same conditions [15, 16].

Based on the LDPC code on finite field $GF(q)$, similar to the construction method for a binary LDPC code, a check matrix H_q on $GF(q)$ is generated through a random Tanner graph. We set the domain element of $H_q = (h_{sk})_{m \times n}$, $h_{sk} = i, i \in 0, 1, \dots, q - 1$. When $i \neq 0$, there is a connected edge with a weight value between the corresponding variable node and the check node in the Tanner graph. The weight value is equal to h_{sk} and each variable node in the figure represents p binary bits.

Figure 4.5a shows a check matrix based on $GF(4)$ and its corresponding Tanner graph. If each field element in the check matrix H_q corresponds to a submatrix of $p \times p$, and the submatrix satisfies the field operation, then the check matrix H_q corresponds to a binary check matrix, as shown in Fig. 4.5b. The bold lines in the figure represent rings of length four. The purpose of constructing LDPC codes based on $GF(q)$ is to eliminate the small loops in random Tanner graphs that seriously affect the convergence of LDPC-code decoding and improve the decoding performance by merging nodes. Moreover, the performance of LDPC codes constructed in larger domains can be greatly improved.

Mackay has proved [17, 18] that for a given decoder, when the column weight (fixed constant) of check matrix H is sufficiently large and the code length is sufficiently large, the performance of LDPC codes can approach the Shannon limit. In other words, the column weight is conducive to fast decoding. However, if the column weight is increased, the number of rings in the corresponding bidirectional graph will increase and the iterative-decoding performance will be degraded.

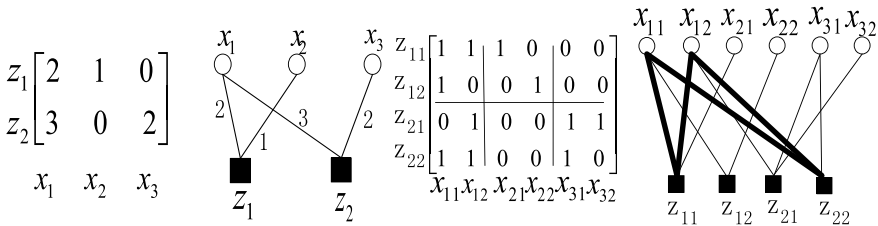


Fig. 4.5 Tanner diagram of a check matrix and its equivalent matrix

An LDPC code constructed on $GF(q)$ can solve this problem by increasing the column weight of its check matrix (that is, increase the column weight of its corresponding binary check matrix); however, the bidirectional graph they decode is the same. In $GF(q)$, the number of rings between nodes will not increase, and the decoding performance will be significantly improved.

(2) Regular and irregular LDPC codes

The LDPC-code checksum matrix constructed by Gallager, whose row weight and column weight are equal, is a regular LDPC code. Irregular LDPC codes are LDPC codes based on irregular random bidirectional graphs. Davey et al. found through the study of LDPC-code performance [11, 12] that the column weight of the check matrix is the main factor determining the performance of an LDPC code, and the row weight has little influence. Therefore, LDPC codes with fixed column weights and unfixed row weights are also regarded as regular LDPC codes.

Studies have shown [17, 18] that irregular LDPC codes constructed by irregular Tanner graphs have better decoding performance than regular LDPC codes, especially in the case of large code lengths. In an AWGN channel, Davey's 16,000-bit irregular code has a gain of about 0.4 dB, compared with the 40,000-bit regular code, when the bit-error rate is 10^{-5} and the bit rate is 1/4.

In an irregular Tanner graph, the numbers of variable nodes and check nodes connected to the side are not fixed. They have distribution sequences $\{\lambda_1, \lambda_2, \dots, \lambda_{d_v}\}$ and $\{\rho_1, \rho_2, \dots, \rho_{d_c}\}$, where λ_i and i denote the information on nodes connected to the edge of the total ratio, ρ_j and the degree of j denote the check nodes connected to the side of the total ratio, and d_v and d_c respectively represent the maximum distribution degree of the variable and parity nodes. The distribution sequences satisfy $\sum_{i=1}^{d_v} \lambda_i = 1$ and $\sum_{j=1}^{d_c} \rho_j = 1$, respectively. The degree distributions of the information nodes and check nodes are defined as

$$\begin{aligned}\lambda(x) &= \sum_{i=1}^{d_v} \lambda_i x^{i-1} \\ \rho(x) &= \sum_{j=1}^{d_c} \rho_j x^{j-1}.\end{aligned}\tag{4.4}$$

Let $\lambda(1) = 1$ and $\rho(1) = 1$. Let the check matrix of an irregular LDPC code be $H_{m \times n}$. According to the degree-distribution function, the number of variable nodes whose degree is i in the Tanner diagram is $n \frac{\lambda_i/i}{\sum_{j \geq 1} \lambda_j/k} = n \frac{\lambda_i/i}{\int_0^1 \lambda(x) dx}$, the number of check nodes whose degree is j is $m \frac{\rho_j/j}{\sum_{i \geq 1} \rho_i/k} = m \frac{\rho_j/j}{\int_0^1 \rho(x) dx}$, and the total number of edges connecting the variable nodes and check nodes is

$$\begin{aligned}
E &= n \sum_{i \geq 1} i \frac{\lambda_i/i}{\int_0^1 \lambda(x)dx} = n \frac{1}{\int_0^1 \lambda(x)dx} \\
&= m \sum_{j \geq 1} j \frac{\rho_j/j}{\int_0^1 \rho(x)dx} = m \frac{1}{\int_0^1 \rho(x)dx}.
\end{aligned} \tag{4.5}$$

The bit rate of irregular LDPC codes constructed by degree-distribution functions $\gamma(x)$ and $\rho(x)$ is

$$r(\lambda, \rho) = \frac{n - m}{n} = 1 - \frac{\int_0^1 \lambda(x)dx}{\int_0^1 \lambda(x)dx}. \tag{4.6}$$

When the check matrix is not full rank, the actual bit rate is slightly larger than $r(\lambda, \rho)$.

Using the characteristics of an irregular Tanner graph, we can explain the reason why an irregular LDPC code is better than a regular LDPC code. In a Tanner diagram, the total number of edges connected to variable and check nodes is equal. From the variable node's perspective, the larger the node degree is, the more information it can obtain from neighboring check nodes; thus, it can judge its correct value more accurately. However, from the check node's perspective, it is the opposite: the smaller the degree, the better. If the degree of the check node is smaller, the more valuable the information it can provide to its neighboring information nodes.

The irregular Tanner diagram clearly has a better balance between the two opposing requirements. If an irregular Tanner graph structure is adopted for LDPC codes, it can quickly obtain the correct value from a large degree of variables; thus, it can provide more accurate information about the probability of check nodes. In addition, these check nodes can give the small degree of variable nodes more effective information.

To produce an avalanche effect, the large degree of the variable node first obtains the correct value. It is then transmitted to the neighboring check nodes, through which the variable nodes with a small degree can obtain the correct information. Therefore, the irregular code can obtain better performance than the regular code.

4.1.4 LDPC-Code Verification-Matrix Construction Method

LDPC codes can be divided into random and algebraic structures, depending on their construction methods. Gallager and Mackay et al. used random methods to construct LDPC codes. The codeword parameters of LDPC codes constructed by random methods are flexible. However, when randomly constructing LDPC codes with high bit rates and medium and short lengths, it is difficult to avoid four-line cycles in a bidirectional graph, which has no certain code structure and high coding complexity.

LDPC codes constructed using an algebraic method have cyclic or quasi-cyclic structures, the coding is simple, and the bidirectional graph corresponding to the codes has no four-line cycles. The LDPC codes constructed using an algebraic method are simulated by confidence-transfer decoding and show good performance in an AWGN channel. The general methods for generating sparse check matrices are as follows [20]:

1. Generate an $(N - K) \times N$ -dimensional all-zero matrix H . To distribute the degrees, rotate the elements randomly for different nodes.
2. Generate an $(N - K) \times N$ -dimensional all-zero matrix, randomly flipping the elements so that the column weight remains d_v .
3. Follow step (2), but try to keep the line weight even, as d_c .
4. Generate a matrix with a fixed column weight of d_v and uniform row weight of d_c , so that the number of rows with the same value between the two columns (called the overlap number) does not exceed 1; that is, in H , there is no four-line cycle in the matrix or Tanner diagram.
5. Generate H according to step (4) and eliminate other short rings.
6. Generate H according to a certain coding structure. For example, the π -rotating LDPC code introduced in Sect. 4.4.1 is generated using a certain structural combination of the permutation matrix to produce H .
7. Generate H according to a certain polynomial.

4.2 Random Construction of LDPC Codes

An LDPC code is completely determined by its check matrix, so the structure of the matrix has a decisive influence on the performance of the code. Different construction methods are used to increase the rings in the graph, to optimize the node distribution of irregular codes, and to reduce the coding complexity.

4.2.1 Gallager's Random Construction Method

Gallager's method can be simply described as follows [4]: Regular check matrices (such as the identity matrix) are constructed in a certain manner. All columns of the matrix are randomly permuted and combined to form a series of regular submatrices, and then, these regular submatrices are combined into the required check matrix.

For a rule code (n, j, k) , the check matrix is divided horizontally into j sub-matrices of the same size, and each column of each sub-matrix contains only a single 1. Let H_0 be the pre-constructed regular matrix, and let $\pi_i(H_0)$ ($i = 1, 2, \dots, j - 1$) be the random column-permutation matrix of H_0 ; then H has the following form:

$$H = \begin{bmatrix} H_0 \\ \pi_1(H_0) \\ \vdots \\ \pi_{j-1}(H_0) \end{bmatrix}. \quad (4.7)$$

For the first child matrix of H_0 , the structure of the method is as follows: In sub-matrix H_0 , the ones form a downward ladder-like arrangement; that is, in matrix H_0 , the first i ($i = 1, 2, \dots, n/k$) number ones in row k are distributed in the first column $(i-1)k + 1$ to $i \times k$; $j-1$ in the other matrix is the first column of the matrix and has a random displacement probability.

From the perspective of linear codes, the LDPC codeword C that satisfies the checksum matrix is the intersection of the codeword C^1, C^2, \dots, C^j that satisfies the submatrix j . The codeword C is shown in Fig. 4.6. Medium and short codes with bit rates of $1/2$ are constructed using the Gallager random construction method. Figure 4.7 shows the performance of sum and product decoding algorithms in an AWGN channel.

Fig. 4.6 Schematic diagram of codeword C

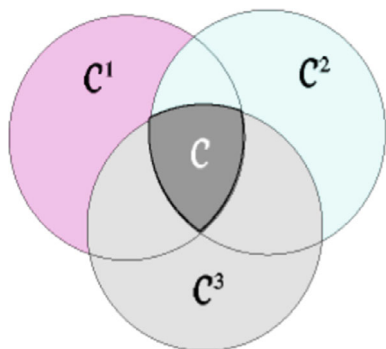
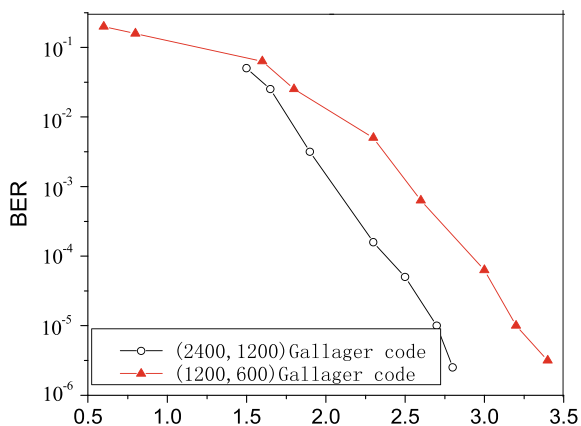


Fig. 4.7 Performance of medium and short codes using the Gallager construction method [14]



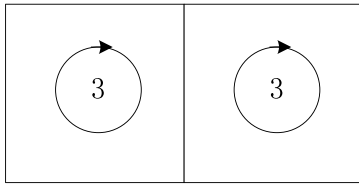
4.2.2 Mackay's Random Construction Method

Mackay's construction is based on bidirectional graphs [17], and his scheme focuses on the elimination of short rings.

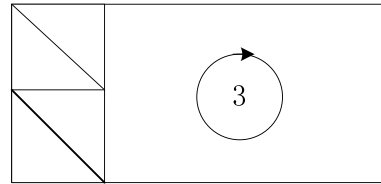
- Construction method 1A: This is the most basic construction method. It constructs the matrix with a fixed column weight of j (e.g., $j = 3$) and a row weight of k , as evenly as possible. The overlapping weight between any two columns is required to be no more than 1. Figure 4.8a shows the constructed LDPC code whose column weight is 3 and row weight is uniformly 6.
- Construction method 2A: Set the weight of the first $m/2$ columns of the check matrix to 2, typically using two $(m/2) \times (m/2)$ identity matrices, stacked vertically. The other parts remain the same as in method 1A. Figure 4.8b shows the codes with a bit rate of $1/3$.
- Construction methods 1B and 2B: Selected partial columns are deleted from the check matrices constructed by methods 1A and 2A, respectively, so that there is no ring (e.g., $l = 6$) smaller than the given length l in the bidirectional graph corresponding to the result matrix.

In addition, Mackay popularized Gallager's random construction method by using the permutation matrix to construct the regular code, as shown in Fig. 4.8c, and constructed a Gallager regular code with a bit rate of $1/2$.

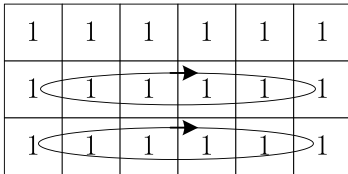
Irregular codes constructed using a permutation matrix can be found in Richardson and Urbanke [18]. The number in the circle in Fig. 4.8 represents the superposition of the permutation matrix with that number in the surrounding matrix, and the diagonal



(a) Construction method 1A



(b) Construction method 2A



(c) Gallager's regular code with bit rate $1/2$

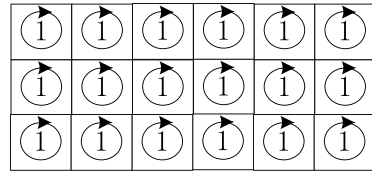
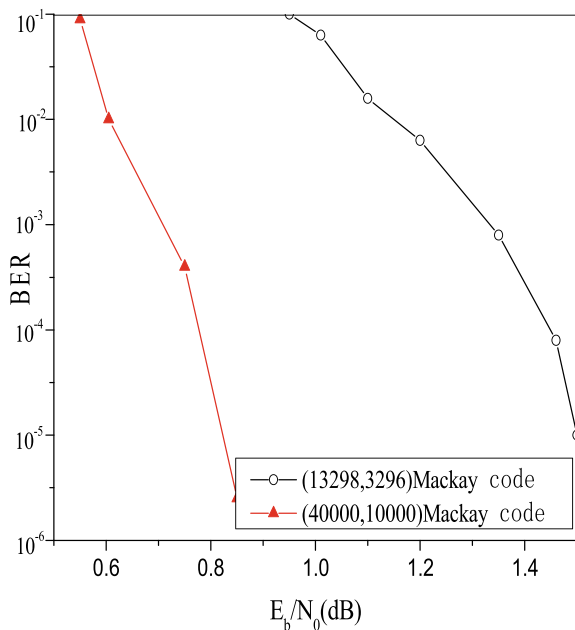


Fig. 4.8 Mackay's construction methods

Fig. 4.9 Performance of Mackay's random construction codes [14]



represents the identity matrix. The rotating ellipse indicates that the columns in the matrix block are free to permute.

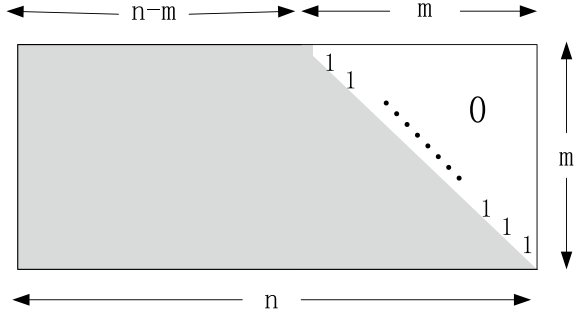
Two codes are designed according to Mackay's random construction method: a long code with a code length of 13,298 and 3296 information bits, and a bit rate of 0.248 and a long code with a code length of 40,000, with 10,000 information bits, and a bit rate of 1/4. The performance of the sum and integration decoding algorithms in an AWGN channel is shown in Fig. 4.9.

4.2.3 RU Random Construction Method

Gallager and Mackay et al. used random methods to construct LDPC codes, whose coding complexity was proportional to the square of the code length. T.J. Richardson and R.L. Urbanke (RU) proposed an effective coding method with a triangular-like check matrix. The encoder complexity of the LDPC code can be controlled within a linear relationship with the code length [19].

(1) Traditional LDPC-code encoding

Let the check matrix H be $m \times n$ -dimensional and full rank. The direct coding method is to transform H into lower triangular form, as shown in Fig. 4.10, through Gaussian elimination. X is divided into two parts, according to the system form; one part is

Fig. 4.10 Check matrix in lower triangular form

information bit S , where $s \in q^{n-m}$, and the other part is parity bit p , where $p \in q^m$. $x = (s, p)$. We follow these steps to construct the system encoder:

1. Take $n - m$ information bits as vector S .
2. Use the backward-iteration method to calculate a check vector of m bits.

$$p_l = \sum_{j=1}^{n-m} H_{l,j} s_j + \sum_{j=1}^{l-1} H_{l,j+n-m} p_j \quad l = 1, 2, \dots, m \quad (4.8)$$

The coding complexity includes two parts: converting H into lower triangular form has a complexity of $O(n^3)$; the other part has a coding complexity of $O(n^2)$. The matrix after H is reduced to lower triangular form is often not sparse. Encoding requires $n^2 r(1-r)/2$ XOR operations, and r is the bit rate. When n is very large, the algorithm is difficult to implement in hardware. However, if H is in lower triangular form and the number of ones in the matrix is proportional to n , the coding complexity can be guaranteed to be linear [21].

(2) RU algorithm

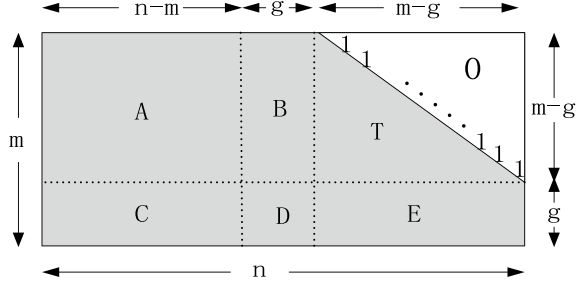
The matrix H is transformed into a subtrigonometric check matrix, as shown in Fig. 4.11 through row and row transformations. At this time, the matrix is still sparse. Suppose the new check matrix is

$$H = \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix}, \quad (4.9)$$

where A, B, C, D, E , and T are $(m-g) \times (n-m)$, $(m-g) \times g$, $g \times (n-m)$, $g \times g$, $g \times (m-g)$, and $(m-g) \times (m-g)$ -dimensional matrices, respectively. All matrices are sparse and T the lower triangular matrix. We multiply matrix H by a matrix to obtain

$$\begin{bmatrix} 1 & 0 \\ -ET^{-1} & I \end{bmatrix} H = \begin{bmatrix} A & B & T \\ -ET^{-1}A + C & -ET^{-1}B + D & 0 \end{bmatrix}. \quad (4.10)$$

Fig. 4.11 The subtrigonometric check matrix



Let $x = (s, p_1, p_2)$, where s information vectors, and p_1 and p_2 jointly define check vectors. p_1 is length g and p_2 is length $m - g$. The following can be obtained from $Hx^T = 0^T$:

$$\begin{aligned} As^T + Bp_1^T + Tp_2^T &= 0 \\ (-ET^{-1}A + C)s^T + (-ET^{-1}B + D)p_1^T &= 0. \end{aligned} \quad (4.11)$$

Let $\varphi = -ET^{-1}B + D$ be reversible and, from Eq. (4.11):

$$\begin{aligned} p_1^T &= -\varphi^{-1}(-ET^{-1}A + C)s^T \\ p_2^T &= -T^{-1}(As^T + Bp_1^T). \end{aligned} \quad (4.12)$$

Finally, the sending vector $x = (s, p_1, p_2)$ is obtained. The complexity of calculation p_1 is $O(n + g^2)$ and the complexity of calculation p_2 is $O(n)$. It can be seen from the above analysis that in the pre-processing to transform matrix H into an equivalent lower triangular matrix, g should be made as small as possible. This is the key to reducing the coding complexity. The greedy algorithm [19, 22] is a better method to reduce g .

The coding complexity of regular codes (3, 6) of code length n is not greater than $0.0172n^2 + O(n)$, because the quadratic term has a very small constant factor. Thus, the complexity of the encoder is controlled within a linear relationship with the code length, making the coding easy to implement.

4.3 Construction of Euclidean Geometric LDPC Codes

LDPC codes have many advantages compared with other codewords; however, when the code length is sufficiently long, the coding complexity will be very large, without using a hardware implementation. However, a check matrix constructed with an algebraic structure has a loop structure and no short loops; hence, the coding complexity is low and it is convenient for a hardware implementation. We will briefly introduce

the basic knowledge of finite fields, and introduce LDPC codes constructed using the Euclidean geometry method.

4.3.1 Basic Theory of Finite Fields

A Euclidean geometric code is a regular code constructed using mathematical tools; its construction method will be given later. First, the concept of Euclidean geometry is presented [23].

Let A be the primitive element of a finite field $GF(2^m)$ and $1, a, \dots, a^{m-1}$ are the basis of $GF(2^m)$. Then, any element a^j in $GF(2^m)$ can be linearly represented by this basis, i.e.,

$$a^j = a_{m-1}^{(j)}a^{m-1} + a_{m-2}^{(j)}a^{m-2} + \dots + a_0^{(j)}. \quad (4.13)$$

In the equation $a_{m-1}^{(j)}, a_{m-2}^{(j)}, \dots, a_0^{(j)} \in GF(2^s)$, if a^j is regarded as a point in a linear space of m dimensions, its coordinate representation is

$$a^j = (a_{m-1}^{(j)}, a_{m-2}^{(j)}, \dots, a_0^{(j)}). \quad (4.14)$$

We define $a^\infty = 0 = (0, 0, \dots, 0)$ as the origin of the m -dimensional linear space; thus, the linear space composed of these 2^m points is called the Euclidean geometric space on $GF(2^m)$. It is referred to as a Euclidean geometric space and denoted by $EG(m, 2^s)$. After removing the origin a^∞ and all lines passing through it from $EG(m, 2^s)$, the subgeometric space $EG^*(m, 2^s)$ of $EG(m, 2^s)$ is obtained.

For an m -dimensional Euclidean geometric space $GF(m, q)$ over a Galois domain $GF(q)$, which contains q^m points and J lines,

$$J \triangleq J_{EG}(m, 1) = q^{m-1}(q^m - 1)/(q - 1). \quad (4.15)$$

An m -vector in a finite field $GF(q)$ corresponds to a point on the Euclidean geometric space $EG(m, q)$, and all m -vector spaces are represented by v . A line on $EG(m, q)$, corresponding to a one-dimensional subspace or coset of vector space V , has q points on each line. Two lines either have no intersection at all or have one and only one intersection.

Let L represent a line on $EG(m, q)$ and p be a point on line L . We can say that line L passes through point p , and if two lines have a common point p , then we can say that they intersect at point p . For any point p on $EG(m, q)$,

$$g \triangleq J_{EG}(m, 1, 0) = (q^m - 1)/(q - 1). \quad (4.16)$$

The line intersects or passes through point p . A finite field $GF(q^m)$, as an extension of $GF(q)$, is an Euclidean geometric space $EG(m, q)$. Assuming α is a primitive of $GF(q^m)$, all powers of α are

$$\alpha^{-\infty} \triangleq 0, \alpha^0 = 1, \alpha, \alpha^2, \dots, \alpha^{q^m-2}. \quad (4.17)$$

All powers of α represent q^m points in Euclidean geometric space $EG(m, q)$, where $\alpha^{-\infty}$ represents the origin of Euclidean geometric space $EG(m, q)$. After removing the origin $\alpha^{-\infty}$ and all lines passing through it, the subgeometric space $EG * (m, q)$ of $EG(m, q)$ contains $q^m - 1$ points that are not the origin.

Definition 4.1 Let a_0, a_1 be two different points in $EG(m, 2^s)$, and $b \in GF(2^s)$ be any element. Then, the set of points shaped like $a_0 + ba_1$ is called a straight line or one-dimensional plane in $EG(m, 2^s)$ [23].

As b is any member of $GF(2^s)$, for each member of b , $a_0 + ba_1$ is a definite point in $EG(m, 2^s)$; therefore, there are 2^s points on line $a_0 + ba_1$.

If $a_0 = 0$, then $\{ba_1 | b \in GF(2^s)\}$ must pass through the origin. If $a_0 \neq 0$, and a_0, a_1 are linearly independent, then $\{a_0 + ba_1 | b \in GF(2^s)\}$ is a line that does not pass through the origin, and this line that does not pass through the origin is parallel to line $\{ba_1 | b \in GF(2^s)\}$ that does pass through the origin. When a_2, a_1 are linearly independent, $\{a_0 + ba_2 | b \in GF(2^s)\}$ and $\{a_0 + ba_1 | b \in GF(2^s)\}$ intersect at a_0 .

By generalizing these concepts, we can define $EG(m, 2^s)$ two-dimensional and dimensional planes in r .

Definition 4.2 If $a_0, a_1, a_2 \in EG(m, 2^s)$ are three points that are not on the same line, that is, they are linearly independent of each other, for $\forall b_1, b_2 \in GF(2^s)$, the form is as follows:

$$a_0 + b_1a_1 + b_2a_2. \quad (4.18)$$

This is called a two-dimensional plane in $EG(m, 2^s)$ [23]. The two-dimensional plane in Eq. (4.18) has 2^{2s} points in total.

Similarly, when $a_0 \neq 0$, the two-dimensional plane passes through the origin, and when $a_0 \neq 0$ and a_0, a_1, a_2 are respectively linearly independent, $a_0 + b_1a_1 + b_2a_2$ is a plane that does not pass through the origin.

In general, the $r + 1$ linearly independent points in $EG(m, 2^s)$ that are not on the $r - 1$ dimensional plane can form an r dimensional plane in $EG(m, 2^s)$ composed of 2^{rs} points:

$$a_0 + b_1a_1 + \dots + b_ra_r. \quad (4.19)$$

The r -dimensional plane is denoted by V_r , because b_1, b_2, \dots, b_r can separately extract any value from the finite field $GF(2^s)$, and r has a total of 2^{rs} points.

4.3.2 Euclidean Geometric LDPC Codes of the First Cyclic-Structure Type

We will construct an EG-LDPC code of the first type of loop [24]. Now, consider an m -dimensional Euclidean geometric subspace $EG^*(m, 2^s)$ defined on $GF(2^s)$ that contains $2^{ms} - 1$ points, each of which is represented by a dimensional vector defined on $GF(2^s)$, with $0 = (0, 0, \dots, 0)$ representing the origin in the space. Let α be a primitive element in a finite field $GF(2^s)$; then, $\alpha^0 = 1, \alpha^2, \alpha^{2^{ms}-2}, \dots$, which can represent all non-zero elements in the Euclidean geometric subspace $EG^*(m, 2^s)$ [25].

$$\text{Let } v = (v_0, v_1, \dots, v_{2^{ms}-2}) \quad (4.20)$$

represent a $(2^{ms} - 1)$ -dimensional vector defined on $EG^*(m, 2^s)$, each element of which corresponds to a $2^{ms} - 1$ non-zero α_j in the Euclidean geometric space $EG^*(m, 2^s)$.

According to Eq. (4.15), the total number of lines in Euclidean geometric subspace $EG^*(m, 2^s)$ is

$$J_0 = \frac{(2^{(m-1)s} - 1)(2^{ms} - 1)}{2^s - 1}. \quad (4.21)$$

Each point is intersected by γ_0 lines, where

$$\gamma_0 = \frac{2^{ms} - 1}{2^s - 1} - 1. \quad (4.22)$$

We construct a check matrix $H_{EG,c}^{(1)}$, where c indicates that the matrix has a cyclic structure. It can be seen from the above that there is a point $n_0 = 2^{sm} - 1$ and a line $J_0 = (2^{(m-1)s} - 1)(2^{ms} - 1)/(2^s - 1)$ that do not pass through the origin in space $EG^*(m, q)$, which are the columns and rows in matrix $H_{EG,c}^{(1)}$ respectively. Because each line in the Euclidean geometric subspace $EG^*(m, q)$ has a total of 2^s points, the line weight of each row in $H_{EG,c}^{(1)}$ is $\rho_0 = 2^s$. Meanwhile, according to Eq. (4.22), the weight of each column of the checksum matrix $H_{EG,c}^{(1)}$ is γ_0 . According to the above, the density of the check matrix $H_{EG,c}^{(1)}$ is r_0 :

$$r_0 = \frac{\rho_0}{n_0} = \frac{\gamma_0}{J_0} = \frac{2^s}{2^{ms} - 1}. \quad (4.23)$$

Therefore, a $(2^{ms} - 1) \times (2^{ms} - 1)$ check matrix is constructed. However, check matrix $H_{EG,c}^{(1)}$ is not necessarily a full-row rank matrix, and most of the time, the number of rows in check matrix $H_{EG,c}^{(1)}$ far exceeds the rank.

Theorem 4.1 If a linear block code is defined by a check matrix, then for each bit in the codeword, there will be γ check equations in the check matrix that are orthogonal to this bit; then, the minimum Hamming distance of this linear block code will have a lower limit [25] that satisfies

$$d_{\min} \geq \gamma + 1. \quad (4.24)$$

Theorem 4.2 The check matrix H constructed in Euclidean geometric space $EG(m, 2^s)$ of all the rows of vectors that do not go through the one-mapping of the origin—but not the columns corresponding to the origin—is defined as a 2^{ms-1} maximum binary cyclic code [25].

The EG-LDPC code constructed from Theorem 4.2 is a cyclic code. Because the check matrix H is the generation matrix of its dual code, it can be known from symmetry that the codeword defined by H will also belong to the cyclic code.

The generation polynomial of a codeword can be obtained from its roots. Let h be a non-negative integer smaller than 2^{ms} then, h can be expressed by Eq. (4.25):

$$h = \delta_0 + \delta_1 2^s + \delta_2 2^{2s} + \delta_{m-1} 2^{(m-1)s}. \quad (4.25)$$

Therefore, for all $0 \leq i \leq m$, we have $0 \leq \delta_i < 2^s$. If the weight of h is expressed by $W_{2^s}(h)$, then

$$W_{2^s}(h) = \sum_{i=0}^{m-1} \delta_i. \quad (4.26)$$

Therefore, the difference of $h - W_{2^s}(h)$ can be expressed as

$$h - W_{2^s}(h) = \delta_1(2^s - 1) + \delta_2(2^{2s} - 1) + \cdots + \delta_{m-1}(2^{2(m-1)s} - 1). \quad (4.27)$$

h must be divisible by $2^s - 1$, if and only if $W_{2^s}(h)$ is divisible by $2^s - 1$. Let $h^{(l)}$ be the remainder of $2^l h$ divided by $2^{ms} - 1$:

$$2^l h = q(2^{ms} - 1) + h^{(l)}, \quad (4.28)$$

where $0 \leq h^{(l)} < 2^{ms} - 1$. From Eq. (4.28), it can be concluded that $h^{(l)}$ must be divisible by $2^s - 1$, if and only if h is divisible by $2^s - 1$. Theorem 4.3 below explains how to obtain the root of the cyclic code defined by the check matrix H , which is proved in reference [24].

Theorem 4.3 Let α be a primitive element of a finite field $GF(2^{ms})$, and let h be a non-negative integer smaller than $2^{ms} - 1$. Then, the generating polynomial of the cyclic code of length $2^{ms} - 1$ defined by the null space of the check matrix $H_{EG,c}^{(1)}$ has roots α^h [25], if and only if

$$0 < \max_{0 \leq l < s} W_{2^s}(h^{(l)}) \leq (m-1)(2^s - 1). \quad (4.29)$$

Let (1) be the minimum integer satisfying Eq. (4.23), which can be proved as follows [26]:

$$\begin{aligned} h_0 &= (2^s - 1) + (2^s - 1)2^s + \cdots + (2^s - 1)2^{(m-3)s} + 2^{(m-2)s} + 2^{(m-1)s} \\ &= 2^{(m-1)s} + 2^{(m-2)s+1} - 1. \end{aligned} \quad (4.30)$$

Therefore, it can be known that $\alpha, \alpha^2, \dots, \alpha^{h_0-1}$ is the root of the generated polynomial. Its minimum Hamming distance can be obtained by the Bose–Chaudhuri–Hocquenghem (BCH) limit [27]:

$$d_{\min} \geq 2^{(m-1)s} + 2^{(m-2)s+1} - 1. \quad (4.31)$$

According to Eq. (4.24), the minimum Hamming distance can also be obtained:

$$d_{\min} \geq \frac{2^{ms} - 1}{2^s - 1}. \quad (4.32)$$

When $m = 2$, it generates a class of EG-LDPC codes with cyclic structures. The check matrix H of the EG-LDPC code of this cycle is a square matrix, and some of its parameters can be calculated [8].

- Yards long: $n = 2^{2s} - 1$
- Queue bit rate: $R = K/n$
- Number of lines in the verification matrix: $J_0 = 2^{2s} - 1$
- Line weight of a verification matrix: $\rho_0 = 2^s$
- Column weight of the verification matrix: $\gamma_0 = 2^s$
- Row rank of the check matrix for a virtual gateway: $n - k = 3^s - 1$
- Number of information bits in the queue codeword: $k = 2^{2s} - 3^s$
- Minimum Hamming distance: $d_{\min} = 2^s + 1$
- Check-matrix density: $r = 2^s / (2^{2s} - 1)$

Because check matrix H is a square matrix with a cyclic structure, we only need to obtain the vector of its first row, and then perform a cyclic shift $2^{2s} - 2$ times to obtain the vectors of each row of all the check matrices. Table 4.1 lists LDPC codes of type-I cyclic structures with different code lengths when $m = 2$ [25].

4.3.3 Euclidean-Geometric LDPC Codes of the Second Cyclic-Structure Type

The second type of LDPC code with a cyclic structure is the dual code of the first type of cyclic LDPC code [30]. Through the check matrix of the first type of cyclic

Table 4.1 EG-LDPC code parameters of different code lengths

S	N	k	d_{\min}	ρ	γ	r	R
2	15	7	5	4	4	0.267	0.466
3	63	37	9	8	8	0.127	0.587
4	255	175	17	16	16	0.0627	0.686
5	1023	781	33	32	32	0.0313	0.763
6	4095	3367	65	64	64	0.01563	0.822

Euclidean geometric code, we can obtain

$$\mathbf{H}_{EG,c}^{(2)} = [\mathbf{H}_{EG,c}^{(1)}]^T, \quad (4.33)$$

where \mathbf{C} indicates that it has a cyclic structure. Because the check matrix $\mathbf{H}_{EG,c}^{(2)}$ is only the transpose of $\mathbf{H}_{EG,c}^{(1)}$, the rows in $\mathbf{H}_{EG,c}^{(2)}$ correspond to all points in the Euclidean geometric space $EG(m, 2^s)$ except the origin, and the columns in $\mathbf{H}_{EG,c}^{(2)}$ correspond to all vectors mapped from the subspace of dimension 1 in the Euclidean geometric space that do not pass through the origin.

The check matrix $\mathbf{H}_{EG,c}^{(2)}$ contains rows $J_0 = 2^{ms} - 1$ and columns $n_0 = (2^{(m-1)s} - 1)/(2^{ms} - 1)/(2^s - 1)$. Matrix $\mathbf{H}_{EG,c}^{(2)}$ is a regular matrix whose row weight is $\rho_0 = (2^{ms} - 1)/(2^s - 1) - 1$ and column weight is $\gamma_0 = 2^s$.

Similarly, because check matrix $\mathbf{H}_{EG,c}^{(1)}$ is a sparse matrix, its transpose matrix $\mathbf{H}_{EG,c}^{(2)}$ is also a sparse matrix. Therefore, check matrix $\mathbf{H}_{EG,c}^{(2)}$ defines LDPC codes with code length $n_0 = (2^{(m-1)s} - 1)(2^{ms} - 1)/(2^s - 1)$, which are called second-type cyclic EG-LDPC codes. Because the transpose operation of the matrix does not change the row rank of the matrix, the row rank of check matrix $\mathbf{H}_{EG,c}^{(2)}$ is the same as the row rank of check matrix $\mathbf{H}_{EG,c}^{(1)}$.

According to Theorem 4.1, the minimum Hamming distance of codes of the second type of EG-LDPC conforms to

$$d_{\min} \geq 2^s + 1. \quad (4.34)$$

The Tanner graphs of check matrix $\mathbf{H}_{EG,c}^{(2)}$ and check matrix $\mathbf{H}_{EG,c}^{(1)}$ are a duality, so there is no ring of length four in the Tanner graph of check matrix $\mathbf{H}_{EG,c}^{(2)}$.

In fact, the EG-LDPC code defined by check matrix $\mathbf{H}_{EG,c}^{(2)}$ is not completely cyclic; however, as long as we transform the check matrix $\mathbf{H}_{EG,c}^{(2)}$ appropriately, we can construct a matrix with a quasi-cyclic structure; therefore, we call it a cyclic LDPC code.

To transform the second type of Euclidean geometric LDPC codes with cyclic structures into quasi-cyclic structures, we divide the mapping vectors of J_0 lines in Euclidean geometric space $EG(m, 2^s)$ that do not pass through the origin into K cyclic classes:

$$K = \frac{2^{(m-1)s} - 1}{2^s - 1}. \quad (4.35)$$

Each loop class has $2^{ms} - 1$ mapping vectors, of which $2^{ms} - 1$ can be obtained by moving any one of them $2^{ms} - 2$ times. For each cyclic class, we can select only one of them to be identified as the mapping-vector representation, while the rest of the mapping-vector representations can be obtained through a cyclic shift [25].

Now, let us construct a matrix H_0 of $(2^{ms} - 1) \times K$, Its k column is the mapping vector representation in k circular classes. When $1 \leq i \leq 2^{ms} - 2$, let H_i be a matrix of $(2^{ms} - 1) \times K$; all its K columns can be obtained by cycling each column of matrix H_0 i times.

Therefore, matrix $H_{EG,qc}^{(2)}$ can be constructed according to the following form:

$$H_{EG,qc}^{(2)} = [H_0, H_1, H_2, \dots, H_{2^{ms}-2}]. \quad (4.36)$$

Then, the EG-LDPC codes with quasi-cyclic structures, defined by check matrix $H_{EG,qc}^{(2)}$ constructed by Eq. (4.36), are obtained, and the shift of K bits for each codeword remains unchanged.

When $m = 2$, because $H_{EG,c}^{(1)}$ is a $(2^{2s} - 1) \times (2^{2s} - 1)$ square matrix, we only need to obtain the vector of its first row, and then carry out a cyclic shift $2^{2s} - 2$ times to obtain the vectors of each row for the entire check matrix. Its column vectors also have the same property. Therefore, if $m = 2$, we can obtain

$$H_{EG,c}^{(1)} = [H_{EG,c}^{(1)}]^T. \quad (4.37)$$

Therefore, $H_{EG,qc}^{(2)} = H_{EG,c}^{(1)}$, and the cyclic EG-LDPC codes defined by the null space of the two check matrices are exactly the same.

4.3.4 Constructing LDPC Codes from Clusters of Parallel Lines in Euclidean Geometry

$J \triangleq J_{EG}(m, 1) = q^{m-1}(q^m - 1)/(q - 1)$ lines in m -dimensional Euclidean geometric space $EG(m, q)$ can be divided into $K_p = (q^m - 1)/(q - 1)$ groups of parallel lines $\mathcal{P}_1(m, 1), \mathcal{P}_2(m, 1), \dots, \mathcal{P}_{K_p}(m, 1)$. Each cluster contains q^{m-1} parallel lines. For a parallel cluster, a line contains all q^m points in $EG(m, q)$, and only one line in the cluster appears at each point. Each parallel cluster contains a line passing through the origin.

For a line L in $EG(m, q)$, whether it passes through the origin or not, define a heavy vector q^m in the field $GF(2)$:

$$v_L = (v_{-\infty}, v_0, v_1, \dots, v_{q^m-2}), \quad (4.38)$$

where the coordinate values (component values) of the vectors correspond to all points in the geometric space $EG(m, q)$. They can be represented by elements $\alpha^{-\infty} \triangleq 0, \alpha^0 = 1, \alpha, \alpha^2, \dots, \alpha^{q-2}$ on a finite field $GF(q^m)$. If and only if α^j is on L , $v_j = 1$; otherwise, $v_j = 0$. This q^m heavy vector is called the type-2 incident vector of line L .

It should be noted here that a type-2 incident vector of the $EG(m, q)$ midline contains a coordinate component value corresponding to the origin of $EG(m, q)$. The vector weight is q . It can be seen that neither coordinate value of the type-2 incident vector of two parallel lines is the same.

For each parallel-line cluster $\mathcal{P}_i(m, 1)$, $1 \leq i \leq K_{\mathcal{P}}$ in $EG(m, q)$, a matrix $H_{\mathcal{P},i}$ of size $q^{m-1} \times q^m$ in the field of $GF(2)$ can be formed by taking the type-2 incident vector of $q^m - 1$ parallel lines as rows. The row and column weights of the matrix are 1 and Q , respectively. For $1 \leq k \leq K_{\mathcal{P}}$, a matrix of size $kq^{m-1} \times q^m$ is constructed as follows:

$$H_{EG,p,k} = \begin{pmatrix} H_{p,1} \\ H_{p,2} \\ \vdots \\ H_{p,k} \end{pmatrix}, \quad (4.39)$$

where the subscript p stands for a parallel bundle of lines. The rows of matrix $H_{EG,p,k}$ correspond to clusters of k parallel lines of $EG(m, q)$, and the row and column weights of the matrix are k and Q , respectively. The null space of matrix $H_{EG,p,k}$ is given a (k, q) regular LDPC code, $\mathcal{C}_{EG,p,k}$, whose code length is a q^m code and minimum distance is at least $K + 1$. For $k = 1, 1, \dots, K_p$, a series of regular LDPC codes with code length q^m with different bit rates and minimum distances can be constructed. The above construction method produces a class of binary regular LDPC codes.

4.4 LDPC-Code Decoding and Performance Estimation

The decoding algorithm in channel coding is an important factor for determining the coding performance in long codes. In general, the decoding complexity of block codes has an exponential relationship with the code length. When the code length increases to a certain extent, the complexity increase will be uncontrollable and even impossible to be applied in practice.

An important reason for the success of LDPC codes is their advantage in decoding, mainly the message-propagation (MP) algorithm based on the Tanner graph. This is an iterative decoding method, which can overcome the huge decoding computation of long block codes and realize completely parallel operations. It has high-speed decoding potential and low hardware-implementation complexity.

Assume that the output symbol set of the channel, that is, the input symbol set of the decoder, is θ . At moment zero, each variable node x_i ($i \in N, N$ is the code length) receives a piece of relevant channel-output information r_i , where r_i is a

random variable whose value is within θ . At each step of the MP-algorithm decoding process, information is passed between the variable nodes and check nodes along the edge of the bidirectional graph.

First, each variable node x_i sends a message with a value in the symbol set M to all connected check nodes z_j . Specifically, at time zero, the variable node x_i sends r_i as its first message (now, $\Theta \subset M$). Each checksum node z_j processes the information it receives and then returns the information within M to all variable nodes connected to it. Then, each variable node x_i processes the information it receives and the related received value r_i , calculates the new reliability information and transmits it to the connected check node. Each iteration of the algorithm is an information-processing cycle: The check nodes process and transmit information, followed by the variable nodes processing and transmitting information.

The MP-algorithm performance is directly related to the quantization of messages transmitted between nodes. If the symbol set transmitted between nodes is binary (0, 1), the MP algorithm set can be equivalent to the tree-decoding algorithm. The performance is best and most complex when the value of the symbol set transmitted between nodes is in the set of real numbers R . At this time, the MP algorithm is a belief-propagation (BP) algorithm.

4.4.1 LDPC Hard-Decision Bit-Flip Decoding

Gallager proposed a bit-flipping algorithm based on confidence propagation. When processing the received signal, the modulator/demodulator (modem) first makes its best judgment of the input symbol of the modulator, and then sends the hard decision to the decoder, which then makes its best judgment of the message input by the coder. This is the concept of hard-decision decoding [5]. This method only applies to a binary symmetric channel (BSC) and can be regarded as a simplified form of the confidence-propagation algorithm.

Let the hard-decision value, $r = \{r_i\}$, $i = 0, 1, \dots, n-1$, of the receiving sequence be adjoint. The algorithm is described as follows:

- (1) The check matrix of the LDPC code $H = [h_0, h_1, \dots, h_{m-1}]^T$, where $h_j = h_{j,0}, h_{j,1}, \dots, h_{j,n-1}$. The j th check equation can be obtained from h_j , as $h_{j,0}r_0 + h_{j,1}r_1 + \dots + h_{j,n-1}r_{n-1}$. The adjoint of the code is

$$s = r \cdot H^T = (s_0, s_1, \dots, s_{m-1}), \quad s_j = r \times h_j = \sum_{i=0}^{n-1} r_i \times h_{j,i} (\text{mod } 2). \quad (4.40)$$

- (2) If $S_j = 0$, the receiving vector satisfies the j th check equation; otherwise, it does not. The number of each code element that does not satisfy the check equation is

$$f = (f_0, f_1, \dots, f_j, \dots, f_{n-1}) = s \cdot H, \quad f_j = \sum_{i=0}^{m-1} s_i \times h_{i,j}. \quad (4.41)$$

- (3) We find the largest element in f :

$$f_j = \max\{f_0, f_1, \dots, f_{n-1}\}, \quad (4.42)$$

and invert the corresponding symbol r_j of f_j to obtain the new sequence r' .

- (4) The new sequence r' is substituted into Eq. (4.40). If $s = 0$, the decoding succeeds; if $s \neq 0$, we judge whether the maximum number of iterations has been reached. If yes, we stop and the decoding fails; if no, we proceed to step (2).

Owing to the random sparsity of the check matrix, very few bits participate in the check equation; therefore, the check equation is either error-free or there is a high probability of a one-bit error. The error can still be corrected, even if more than one error occurs.

To correct any bit d , a parity tree can be introduced, as shown in Fig. 4.12. d is represented by the nodes at the root of the tree. Each line ascending from the root of the tree represents a parity constraint in the parity set containing d . The other bits in the parity set are represented by nodes at the first level of the tree. Lines ascending from the first layer of the tree to the second layer represent other parity sets containing the first layer's numbers, while nodes in the second layer represent other bits in these parity sets, and so forth.

The parity-tree decoding shown in Fig. 4.12 is carried out in order from outer layer to root. Given an error in bit d and layer 1 e , the first decoding uses layer two's error-free bits and their parity constraints to correct the layer-one error bits. The second decoding uses the correct number in the first layer to correct the number d .

This algorithm only needs to calculate the checksum of modulo 2, and has low computational complexity. It mainly adopts XOR gates and a comparator, which is easy to complete in hardware; however, its performance is not as good as that of the BP algorithm. A (3, 6) regular LDPC code with code length $N = 1008$ and information-bit length $K = 504$ is used to modulate with binary phase-shift keying (BPSK) in an AWGN channel. Figure 4.13 shows the hard-decision decoding result.

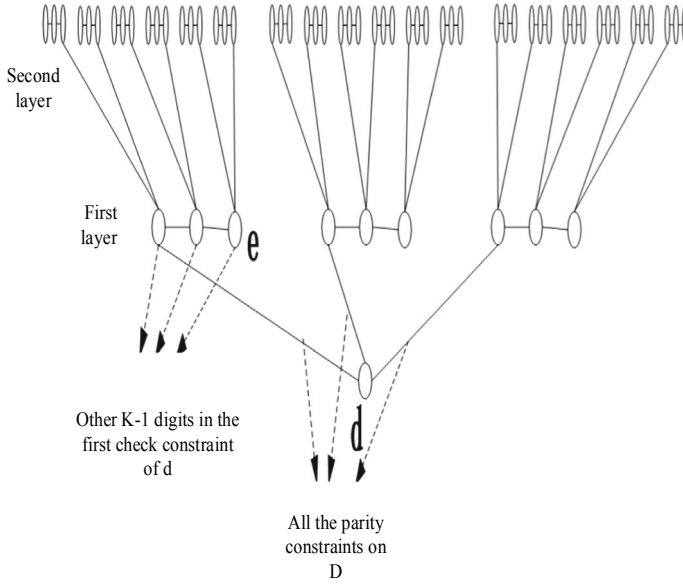
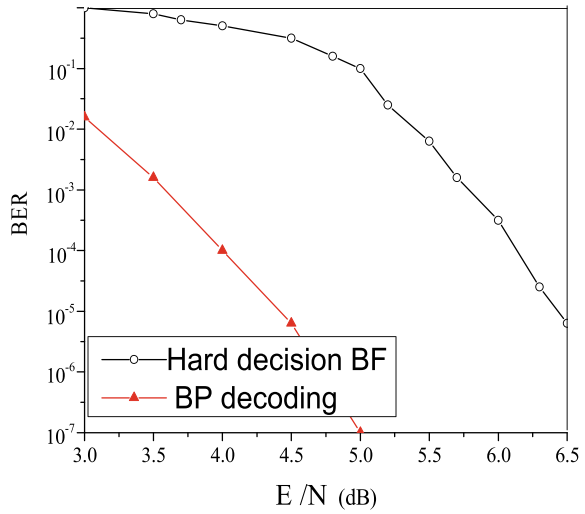


Fig. 4.12 Verification-set tree

Fig. 4.13 Hard-decision decoding performance of BPSK modulation in an AWGN channel [14]



4.4.2 BP Iterative-Decoding Algorithm for LDPC Codes

(1) Gallager probabilistic decoding algorithm

Lemma 4.1 [4, 5]: Assume that m has an independent binary number $a = (a_1 a_2 \dots a_m)$, where $P_r(a_k = 1) = p_k$ represents the probability that bit k is 1; then,

the probability that a has an even number of ones is $P_e(m) = [1 + \prod_{k=1}^m (1 - 2p_k)]/2$, and the probability of an odd number of ones is $P_o(m) = [1 - \prod_{k=1}^m (1 - 2p_k)]/2$.

We set (n, j, k) of length n as the parity matrix H of the LDPC code, as shown in Fig. 4.12. The received set of real vectors is denoted as $\{y\}$, the event that the transmitted information bits X satisfy all the check equations containing x_d is denoted as S , and the conditional probability of bits $x_d = 1$ (or $x_d = 0$) regarding $\{y\}$ and S is $P_r(x_d = 1|\{y\}, S)$ (or $P_r(x_d = 0|\{y\}, S)$). The definition of conditional probability is

$$\frac{P_r(x_d = 0|\{y\}, S)}{P_r(x_d = 1|\{y\}, S)} = \frac{1 - P_d}{P_d} \prod_{i=1}^j \frac{P_r(S|x_d = 0, \{y\})}{P_r(S|x_d = 1, \{y\})}, \quad (4.43)$$

where P_d represents the probability that x_d in the codeword obtained through channel characteristics is 1.

When $x_d = 0$, an equation containing x_d has an even number of ones in other ratios; then, the modulo-2 sum of the entire check equation is zero; that is, this check equation satisfies. Because each bit is statistically independent, the probability that all the checksum equations including x_d satisfy is the product of the probability that each checksum equation satisfies; then,

$$P_r(S|x_d = 0, \{y\}) = \prod_{i=1}^j \frac{1 + \prod_{l=1}^{k-1} (1 - 2P_{il})}{2}, \quad (4.44)$$

where P_{il} represents the probability that the l th bit in the i th check equation containing x_d is 1. Similarly, when $x_d = 1$,

$$P_r(S|x_d = 1, \{y\}) = \prod_{i=1}^j \frac{1 - \prod_{l=1}^{k-1} (1 - 2P_{il})}{2}. \quad (4.45)$$

We substitute Eqs. (4.44) and (4.45) into Eq. (4.43) to obtain

$$\frac{P_r(x_d = 0|\{y\}, S)}{P_r(x_d = 1|\{y\}, S)} = \frac{1 - P_d}{P_d} \prod_{i=1}^j \left[\frac{1 + \prod_{l=1}^{k-1} (1 - 2P_{il})}{1 - \prod_{l=1}^{k-1} (1 - 2P_{il})} \right]. \quad (4.46)$$

It is difficult to directly calculate using Eq. (4.46); therefore, it is converted into log-likelihood ratio (LLR) form for convenient operation [33].

Definition

$$\begin{aligned}
\ln\left(\frac{1 - P_d}{P_d}\right) &= \alpha_d \beta_d \\
\ln\left(\frac{1 - P_{il}}{P_{il}}\right) &= \alpha_{il} \beta_{il} \\
\ln\left(\frac{P_r(x_d = 0|\{y\}, S)}{P_r(x_d = 1|\{y\}, S)}\right) &= \alpha_d' \beta_d',
\end{aligned} \tag{4.47}$$

where α is the sign of the log value and β is the absolute value of the log value. We substitute into Eq. (4.46) to obtain

$$\begin{aligned}
\alpha_d' \beta_d' &= \ln\left(\frac{P_r[x_d = 0|\{y\}, S]}{P_r[x_d = 1|\{y\}, S]}\right) \\
&= \ln\left(\frac{1 - P_d}{P_d} \prod_{i=1}^j \left[\frac{1 + \prod_{l=1}^{k-1} (1 - 2P_{il})}{1 - \prod_{l=1}^{k-1} (1 - 2P_{il})} \right]\right) \\
&= \alpha_d \beta_d + \sum_{i=1}^j \left\{ \left(\prod_{l=1}^{k-1} \alpha_{il} \right) f\left[\sum_{l=1}^{k-1} f(\beta_{il}) \right] \right\}.
\end{aligned} \tag{4.48}$$

Among them, $f(\beta) = \ln\left(\frac{e^\beta + 1}{e^\beta - 1}\right) = -\ln\left[\tanh\left(\frac{\beta}{2}\right)\right]$. $f(\beta)$ can be achieved by establishing look-up tables and $f(\beta) = f^{-1}(\beta)$

(2) BP decoding method in the probability domain

Lemma 4.2 [2, 34]: Assume $y_i = x_i + n_i$, where n_i satisfies a Gaussian random variable with a $(0, \sigma^2)$ distribution, and $P_r(x_i = 1) = P_r(x_i = 0) = 0.5$; then, $P_r(x_i = x|y_i) = 1/(1 + e^{-2xy_i/\sigma^2})$ ($\forall x \in \{-1, 1\}$).

Let $C(i)$ represent the set of check nodes adjacent to variable node x_i , and $C(i) \setminus j$ represent the set of j check nodes removed from $C(i)$; $R(j)$ represents the set of variable nodes adjacent to check node z_j , and $R(j) \setminus i$ represents the set of i variable nodes removed from $R(j)$.

Definition $q_{ij}(b)$ is the soft information passed from variable node x_i to check node z_j . It represents the probability that $x_i = b$ (b taking 0 or 1) under the condition that all check nodes adjacent to it, except the j th check node, provide external information, given y_i . $r_{ji}(b)$ is the external information transferred from check node z_j to variable node x_i , and represents the probability of establishing a check equation under the condition that $x_i = b$ and the other bits of the j th check equation satisfy the probability $q_{i'j}(i \neq i')$.

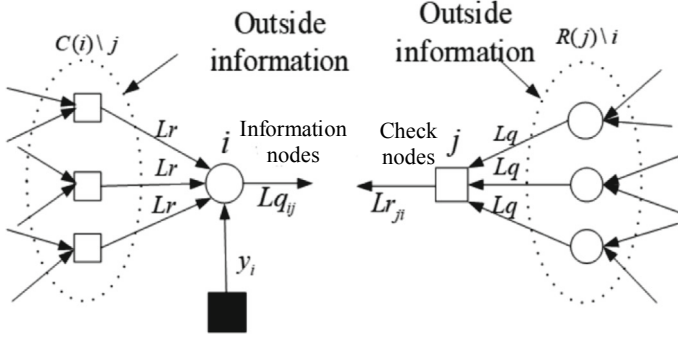


Fig. 4.14 Message-propagation diagram of the BP decoding algorithm

The value of the symbol set transmitted between decoding nodes in the probability domain is in the real number set R , which has the best performance and is also the most complex. The MP algorithm is a confidence-propagation algorithm. Figure 4.14 shows the information transmission of the decoding algorithm.

Probabilistic domain decoding is performed through defined iterations between q_{ij} and r_{ji} .

$$r_{ji}(0) = \frac{1}{2} + \frac{1}{2} \prod_{i' \in R(j) \setminus i} (1 - 2q_{i'j}(1)). \quad (4.49)$$

$$r_{ji}(1) = \frac{1}{2} - \frac{1}{2} \prod_{i' \in R(j) \setminus i} (1 - 2q_{i'j}(1)) \quad (4.50)$$

Thus, Eq. (4.46) is transformed into

$$\frac{P_r(x_i = 0 | \{y\}, S)}{P_r(x_i = 1 | \{y\}, S)} = \frac{1 - P_d}{P_d} \frac{\prod_{j \in C(i)} r_{ji}(0)}{\prod_{j \in C(i)} r_{ji}(1)}. \quad (4.51)$$

$$q_{ij}(0) = K_{ij}(1 - P_i) \prod_{j' \in C(i) \setminus j} r_{j'i}(0) \quad (4.52)$$

$$q_{ij}(1) = K_{ij}P_i \prod_{j' \in C(i) \setminus j} r_{j'i}(1), \quad (4.53)$$

where K_{ij} is to ensure the normalization coefficient of $q_{ij}(0) + q_{ij}(1) = 1$. Calculate after each iteration between q_{ij} and r_{ji} :

$$Q_i(0) = K_i(1 - P_i) \prod_{j \in C(i)} r_{ji}(0) \quad (4.54)$$

$$Q_i(1) = K_i P_i \prod_{j \in C(i)} r_{ji}(1) \quad (4.55)$$

where K_i is the normalization coefficient. In addition, to ensure $Q_i(0) + Q_i(1) = 1$, the decoding steps of the BP algorithm in the probability domain are as follows:

1. Initialization: For $H_{ij} \neq 0$, $q_{ij}(1) = 1 - q_{ij}(0) = P_r^1$, number of iterations $k = 1$.
2. Iterative calculation:
 - a. Calculate r_{ji}^k ($\forall i, j : h_{ji} = 1$) from Eqs. (4.49) and (4.50), where k is the number of iterations.
 - b. Calculate q_{ij}^k ($\forall i, j : h_{ji} = 1$) from Eqs. (4.52) and (4.53), where k is the number of iterations.
3. Decoding:

$Q_i(0)$ and $Q_i(1)$ are calculated according to Eqs. (4.54) and (4.55), when a hard judgment $Q_i(1) \geq 0.5$ is made; otherwise, $x_i = 1$. The resulting decoding result sequence $x_i = 0$ is left multiplied by the check matrix \hat{x}^k to obtain the adjoint vector H .
4. Output the decoding result when $s^k = 0$; otherwise, skip to step 2, or stop when the number of iterations reaches the maximum.

For an iterative calculation, the forward/backward algorithm can be used to simplify appropriately [7].

Let $\delta q_{ij} = q_{ij}(0) - q_{ij}(1)$; then,

$$\delta r_{ji} = r_{ji}(0) - r_{ji}(1) = \prod_{i' \in R(j) \setminus i} \delta q_{i'j}. \quad (4.56)$$

From $r_{ji}(0) + r_{ji}(1) = 1$ comes $r_{ji}(0) = \frac{1}{2}(1 + \delta r_{ji})$ and $r_{ji}(1) = \frac{1}{2}(1 - \delta r_{ji})$. Then,

$$q_{ij}(0) = \alpha_{ij} P_r^0 \prod_{j' \in C(i) \setminus j} r_{j'i}(0) \quad q_{ij}(1) = \alpha_{ij} P_r^1 \prod_{j' \in C(i) \setminus j} r_{j'i}(1), \quad (4.57)$$

where α_{ij} is the normalization coefficient, to ensure $q_{ij}(0) + q_{ij}(1) = 1$.

After each iteration q_{ij} and r_{ji} ,

$$Q_i(0) = \alpha_i (1 - P_i) \prod_{j \in C(i)} r_{ji}(0) \quad Q_i(1) = \alpha_i P_i \prod_{j \in C(i)} r_{ji}(1). \quad (4.58)$$

(3) Logarithmic-domain BP decoding algorithm

BP decoding in the probability domain transmits some probability information between nodes, which requires extensive multiplication, division and logarithmic operations; therefore, it is not suitable for a hardware implementation. The log-likelihood ratio (LLR) is used to transfer information between nodes, which changes

many operations into addition operations, reduces the operation time, and improves the decoding speed.

We define the logarithmic likelihood ratio as follows:

$$L(r_{ji}) = \ln\left(\frac{r_{ji}(0)}{r_{ji}(1)}\right), \quad L(q_{ij}) = \ln\left(\frac{q_{ij}(0)}{q_{ij}(1)}\right), \quad (4.59)$$

$$L(P_i) = \ln\left(\frac{P_r(x_i = 1|y_i)}{P_r(x_i = -1|y_i)}\right) = \frac{2y_i}{\sigma^2}, \quad L(Q_i) = \ln\left(\frac{Q_i(0)}{Q_i(1)}\right). \quad (4.60)$$

Because $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$, if $x = \frac{1}{2} \ln\left(\frac{p_0}{p_1}\right)$, $p_0 + p_1 = 1$ then,

$$\tanh\left(\frac{1}{2} \ln\left(\frac{p_0}{p_1}\right)\right) = p_0 - p_1 = 1 - 2p_1. \quad (4.61)$$

According to Eqs. (4.51)–(4.53), we can obtain

$$\tanh\left(\frac{1}{2}L(r_{ji})\right) = \prod_{i' \in R(j) \setminus i} (q_{i'j}(0) - q_{i'j}(1)) = \prod_{i' \in R(j) \setminus i} \tanh\left(\frac{1}{2}L(q_{i'j})\right). \quad (4.62)$$

Therefore,

$$L(r_{ji}) = 2 \tanh^{-1}\left(\prod_{i' \in R(j) \setminus i} \tanh\left(\frac{1}{2}L(q_{i'j})\right)\right). \quad (4.63)$$

Equation (4.63) has many multiplication operations. Therefore, we redefine it:

$$\begin{aligned} L(q_{i'j}) &= \alpha_{i'j} \beta_{i'j} \\ \alpha_{i'j} &= \text{sign}(L(q_{i'j})), \\ \beta_{i'j} &= |L(q_{i'j})| \end{aligned} \quad (4.64)$$

where $\text{sign}(x) = \begin{cases} 1 & x > 0 \\ -1 & x < 0 \end{cases}$. When $x = 0$, $\text{sign}(x)$ with probability 1/2 is equal to 1 or -1 . According to Lemma 4.2,

$$f(f(x)) = \ln\left[\frac{e^{f(x)} + 1}{e^{f(x)} - 1}\right] = \ln\left(\frac{e^{\ln\frac{e^x+1}{e^x-1}} + 1}{e^{\ln\frac{e^x+1}{e^x-1}} - 1}\right) = \ln\left(\frac{\frac{e^x+1}{e^x-1} + 1}{\frac{e^x+1}{e^x-1} - 1}\right) = x. \quad (4.65)$$

That is, $f^{-1}(x) = f(x)$. Therefore, Eq. (4.54) can be converted into

$$\begin{aligned}
L(r_{ji}) &= 2 \left(\prod_{i' \in R(j) \setminus i} \alpha_{i'j} \right) \times \tanh^{-1} \left(\ln^{-1} \left(\sum_{i' \in R(j) \setminus i} \ln \left(\tanh \left(\frac{1}{2} \beta_{i'j} \right) \right) \right) \right) \\
&= \left(\prod_{i' \in R(j) \setminus i} \alpha_{i'j} \right) \times f \left(\sum_{i' \in R(j) \setminus i} f(\beta_{i'j}) \right).
\end{aligned} \tag{4.66}$$

Thus, Eq. (4.66) can be completed using only XOR and addition operations. The corresponding Eqs. (4.52) and (4.53) can be directly divided, and the logarithmic-likelihood information can be obtained by taking the natural logarithms of both sides:

$$L(q_{ij}) = L(P_i) + \sum_{j' \in C(i) \setminus j} L(r_{ji}). \tag{4.67}$$

Corresponding to Eqs. (4.63) and (4.64), we can obtain

$$L(Q_i) = L(P_i) + \sum_{j \in C(i)} L(r_{ji}). \tag{4.68}$$

It can be seen that Eq. (4.63) mainly adopts product operations, and Eq. (4.67) mainly adopts sum operations; therefore, the BP decoding algorithm is also called the sum-product algorithm.

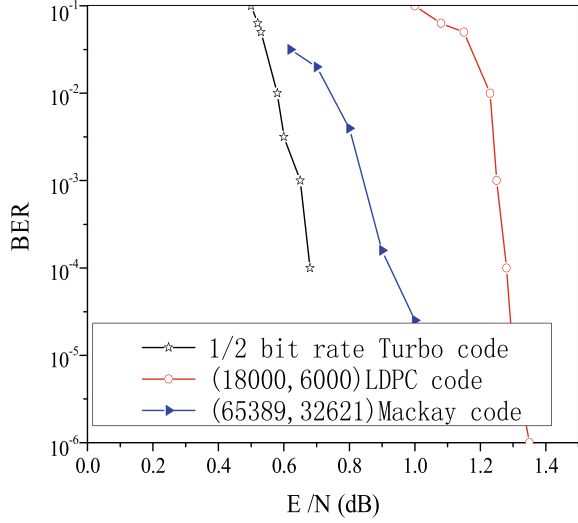
The specific steps of the BP algorithm in the logarithmic domain are as follows:

1. Initialization: $L(q_{ij}^0) = L(P_i)$ and number of iterations $k = 1$.
2. Iterative calculation:
 - (a) The information passed from the check node to the variable node is calculated using Eq. (4.66), $L(r_{ji}^k)$.
 - (b) The information passed to the parity node by the variable node is calculated using Eq. (4.67), $L(q_{ji}^k)$.
3. Decoding:

Equation (4.68) is used to calculate $L(Q_i)$ and make a hard judgment. When $L(Q_i) > 0$, it corresponds to $x_i = 0$; otherwise, $x_i = 1$. The resulting decoding-result sequence \hat{x}^k is left multiplied by the check matrix H to obtain the adjoint vector $s^k = (s_1^k, s_2^k, \dots, s_M^k)$.
4. Output the decoding result, when $S^k = 0$; otherwise, return to step 2, or stop when the number of iterations reaches the maximum.

Figure 4.15 shows the performance of the Mackay and Gallager codes under a Gaussian channel by adopting turbo codes and sum-and-product decoding from reference [35]. The Mackay code length is 65389, with an information-bit length of 32,621, and a bit rate of 0.499. The Gallager code length is 18000, the information-bit length is 6000, and the bit rate is one-third.

Fig. 4.15 Performance comparison of codes with integration-decoded codes [14]



(4) Minimum-sum decoding

In the logarithmic-domain BP algorithm, the auxiliary function $f(x)$ is used. When $x > 0$, the curve of the function is shown in Fig. 4.16. It can be seen that the function decreases rapidly with the increase of the independent variable. Thus,

$$f\left(\sum_{i' \in R(j) \setminus i} f(\beta_{i'j})\right) \approx f\left(f \min_{i' \in R(j) \setminus i} (\beta_{i'j})\right) = \min_{i' \in R(j) \setminus i} (\beta_{i'j}). \quad (4.69)$$

Therefore, Eq. (4.55) can be simplified as

$$L(r_{ji}) = \left(\prod_{i' \in R(j) \setminus i} \alpha_{i'j} \right) \times \min_{i' \in R(j) \setminus i} (\beta_{i'j}). \quad (4.70)$$

Replacing Eq. (4.66) with simplified Eq. (4.70) can not only reduce the complexity and storage of logarithmic-domain decoding, but also reduce the difficulty of fixed-point algorithms. This simplified decoding method is called the minimum-sum (min-sum) algorithm. Figure 4.17 compares the performance of Mackay's randomly constructed LDPC codes with a code length of 13,298 and an information-bit length of 3296 in the logarithmic domain with a minimum-sum decoding algorithm.

Fig. 4.16 Graphs of auxiliary functions [14]

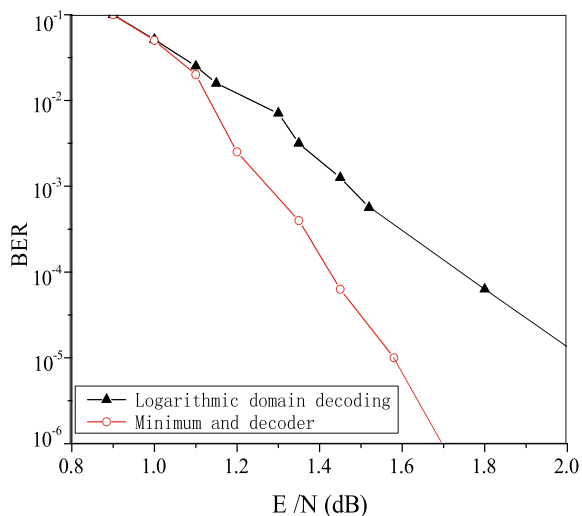
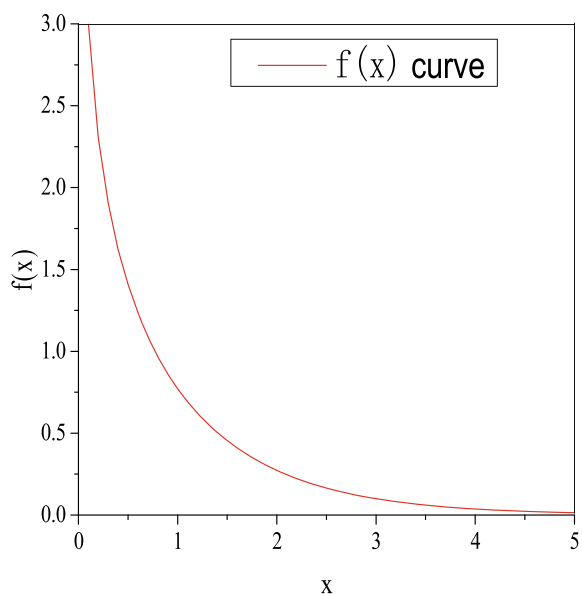


Fig. 4.17 Comparison of the decoding performance of Mackay's random construction codes [14]



4.4.3 Performance Estimation and Analysis of LDPC Codes

(1) Decoding-error probability

In different channel environments, the decoding-error probability may be different when using probabilistic decoding. The following is a simple analysis of a binary symmetric channel (BSC), and a decoding-error upper limit is given.

We set the BSC transfer probability as p_0 and the LDPC code parameter as (n, j, k) . Assume that there are m independent layers in the tree structure in Fig. 4.12, and mark the uppermost layer as layer 0 and the root node as layer m .

The decoding process is as follows: If the $j - 1$ check equations starting from a node in the first layer are not satisfied, the node is flipped; then, the nodes in the second layer are corrected using the nodes in the first layer after error correction, to reach the root node. The decoding process of d is based on the relevant nodes of layer m , while the probability decoding also makes use of the relevant nodes of layer m after m iterations. It is obvious that probability decoding has the most accurate judgment. This bit-flipping algorithm error can be approximated as an upper limit of probabilistic-decoding errors.

The probability of a mistransmission in a node in the first layer is p_0 . In an equation starting from this node to layer 0, if $j - 1$ other nodes still have an even number of errors, the entire check equation will not be satisfied. If $j - 1$ check equations are not satisfied, then this node will be corrected with the following probability:

$$p_0 \left(\frac{1 + (1 - 2p_0)^{k-1}}{2} \right)^{j-1}. \quad (4.71)$$

When this node receives correctly, the check equation caused by other nodes at the relevant layer 0 is not satisfied; thus, the probability of this node being corrected is

$$(1 - p_0) \left(\frac{1 - (1 - 2p_0)^{k-1}}{2} \right)^{j-1}. \quad (4.72)$$

After this decoding process, the probability of a node error in the first layer is

$$p_1 = p_0 - p_0 \left(\frac{1 + (1 - 2p_0)^{k-1}}{2} \right)^{j-1} + (1 - p_0) \left(\frac{1 - (1 - 2p_0)^{k-1}}{2} \right)^{j-1}. \quad (4.73)$$

By analogy, if the error probability of layer i is p_i , then the error probability of a node at layer $i + 1$ is:

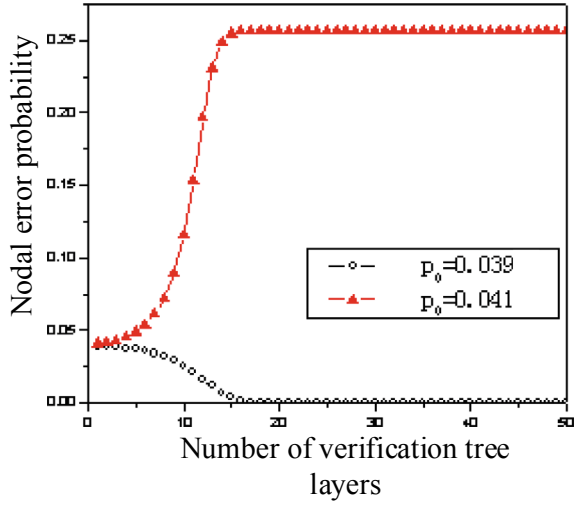
$$p_{i+1} = p_i - p_i \left(\frac{1 + (1 - 2p_i)^{k-1}}{2} \right)^{j-1} + (1 - p_i) \left(\frac{1 - (1 - 2p_i)^{k-1}}{2} \right)^{j-1}. \quad (4.74)$$

When p_0 is sufficiently small, p_i will converge to 0 as i increases. For the actual channel and the determined code parameters, p_i is guaranteed to converge to a p_0 of 0 within a certain range.

Figure 4.18 shows the variation trend of the node-error probability of a Gallager code (3, 6) with different values of Gallager code p_0 . The above algorithm is called Gallager decoding algorithm A [11].

Fig. 4.18 (3, 6)

Gallager-code node-error probability changing with p_0



In fact, when $j > 3$, parameter b can be appropriately selected. When b or more correlation-verification equations are not satisfied, the bit can be flipped. This improved algorithm is called the Gallager decoding algorithm B. The error probability of a node at layer $i + 1$ is [11]

$$\begin{aligned}
 p_{i+1} = & p_i - p_i \sum_{l=b}^{j-1} \binom{j-1}{l} \left(\frac{1 + (1 - 2p_i)^{k-1}}{2} \right)^l \left(\frac{1 - (1 - 2p_i)^{k-1}}{2} \right)^{j-1-l} \\
 & + (1 - p_i) \sum_{l=b}^{j-1} \binom{j-1}{l} \left(\frac{1 - (1 - 2p_i)^{k-1}}{2} \right)^l \left(\frac{1 + (1 - 2p_i)^{k-1}}{2} \right)^{j-1-l}.
 \end{aligned} \quad (4.75)$$

Parameter b should be chosen to minimize p_{i+1} , which can be obtained from the smallest integer satisfying the following inequality:

$$\frac{(1 - p_0)}{p_0} \geq \left(\frac{1 + (1 - 2p_i)^{k-1}}{1 - (1 - 2p_i)^{k-1}} \right)^{2b-j+1}. \quad (4.76)$$

(2) Density evolution theory

It is difficult to theoretically analyze the performance of LDPC codes. At present, the most important thing is that Richardson et al. used the density evolution theory to discuss the asymptotic average performance of LDPC codes [11, 36, 37]. BP decoding makes the message of the codeword bits evolve in the correct direction through message iteration, so that the decoding converges to the correct codeword. Richardson et al. studied messages sent by variable nodes and check nodes on an

acyclic Tanner graph, analyzed the evolution of the message probability-density function in the process of iterative decoding, and established the theory of density evolution.

In a message-transmission iteration evolution, a fixed point, namely, the decoding of the LDPC code, has a fixed threshold. When the signal-to-noise ratio is greater than the threshold, the average error rate of the codeword always tends to zero as the number of iterations and the yard lengths increase. If the signal-to-noise ratio is less than the fixed threshold, then, no matter how long the code or how many times it is iteratively decoded, the average error rate is always greater than some constant. The lower the threshold, the higher the transmission reliability of this code, and the stronger the ability to withstand noise.

At present, there is no deterministic theoretical-derivation method for the degree-distribution design of irregular LDPC codes; however, density evolution theory can be used to measure the progressive average performance of LDPC codes and search for an optimal degree-distribution pair, to optimize the design of LDPC codes.

Chung [38], Hu and Zhang [39] proposed the Gaussian estimation principle to simplify the decoding analysis of LDPC codes on an AWGN channel on the basis of the density evolution theory. Without sacrificing too much accuracy, the calculation of the decoding threshold is simplified from the density-evolution theory model of a multi-dynamic system to a Gaussian estimation model of a single-parameter dynamic system. It can effectively simplify the decoding analysis of LDPC codes and the search for the optimal degree sequence of irregular codes.

a. Theoretical basis of density evolution

In iteration l , in a regular LDPC code of type (n, j, k) , the information sent by check node z to adjacent variable node x is a function of the input information v_1, v_2, \dots, v_{k-1} of the $k - 1$ th adjacent variable node, denoted as $u = \phi_c^{(l)}(v_1, v_2, \dots, v_{k-1})$, where l represents the number of iterations. Similarly, the information sent by variable node x to check node z is a function of the input information u_1, u_2, \dots, u_{j-1} from $j - 1$ adjacent check nodes, denoted as $v = \phi_v^{(l)}(u_0, u_1, \dots, u_{j-1})$, where u_0 is the initial information of the variable node obtained from the channel received value.

To use density evolution theory, two preconditions must be satisfied; one is an independent condition and the other is a symmetric condition. The independence condition is that in the iterative-decoding process, the messages obtained by the variable nodes and check nodes on the Tanner graph are independent of each other, and the BP algorithm on an acyclic graph satisfies the independence of the information. When the loop graph does not contain a loop of path length $L \leq 2l(n)$,

$$l(n) = \frac{\ln n - \ln\left(\frac{jk-j-k}{2k}\right)}{\ln[(j-1)(k-1)]}. \quad (4.77)$$

After decoding l iterations, the independence of the information obtained by each variable node can be guaranteed. When the symmetry condition is satisfied, let $p_e^{(l)}(x)$

be the error probability of codeword x after decoding for l iterations; then, $p_e^{(l)}(x)$ is independent of x , and the decoding-error probability is independent of the transmitted codeword.

According to Eq. (4.68), variable message v in the sum-and-product decoding algorithm is expressed as

$$v = \phi_v^{(l)}(u_0, u_1, \dots, u_{j-1}) = \sum_{i=0}^{j-1} u_i, \quad (4.78)$$

where u_0 represents the LLR value of the initial information, and u_1, u_2, \dots, u_{j-1} represents the check message received by the variable node from adjacent check nodes. Therefore, the probability density of v can be obtained as

$$\tilde{\phi}_v^{(l)}(u_0, u_1, \dots, u_{j-1}) = P_0 \otimes P_1 \otimes \dots \otimes P_{j-1}, \quad (4.79)$$

where P_i represents the probability density of information u_i ($i = 0, 1, \dots, j-1$), and \otimes represents the convolution operation. We use F to represent the Fourier transform; then, Eq. (4.79) can be transformed into

$$\tilde{\phi}_v^{(l)}(u_0, u_1, \dots, u_{j-1}) = F^{-1}(F(P_0)F(P_1) \dots F(P_{j-1})). \quad (4.80)$$

A Fourier transform can simplify the probability-density calculation of v . To effectively calculate Eq. (4.80), a fast Fourier transform (FFT) can also be used.

According to Eq. (4.76), the LLR of the check information sent by check node z to variable node x can be obtained:

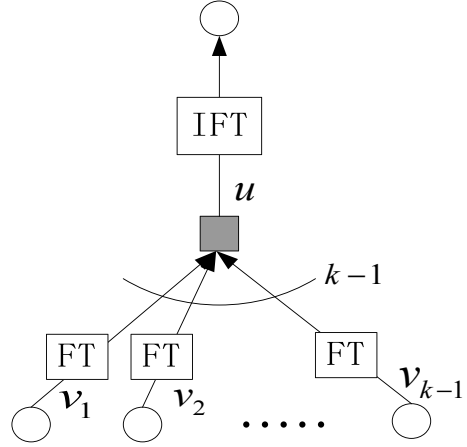
$$\tanh\left(\frac{u}{2}\right) = \prod_{i=1}^{k-1} \tanh\left(\frac{v_i}{2}\right). \quad (4.81)$$

To calculate the probability density of message u using the Fourier transform, the product operation needs to be transformed into a sum operation. We take the logarithm of both sides of Eq. (4.81); however, the domain of $\tanh(x)$ is $(-\infty, +\infty)$ and is a monotonic function. To transform the domain into $[0, +\infty)$, we define a function whose range is on $GF(2)$:

$$\lg \operatorname{sgn}(x) = \begin{cases} 0 & x > 0 \\ 1 & x < 0 \end{cases}. \quad (4.82)$$

$\lg \operatorname{sgn}(x)$ has the probability of 1/2 equal to 0 or 1. Then, we take the logarithm of both sides of Eq. (4.81) and obtain [35]:

Fig. 4.19 Transformation representation of node correction function



$$\left(\lg \operatorname{sgn} \left(\tanh \frac{u}{2} \right), -\log \left| \tanh \frac{u}{2} \right| \right) = \left(\sum_{i=1}^{k-1} \lg \operatorname{sgn} \left(\tanh \frac{v_i}{2} \right), -\sum_{i=1}^{k-1} \log \left| \tanh \frac{v_i}{2} \right| \right). \quad (4.83)$$

The first component on the right side of the equation is added by modulo 2. Thus, the calculation of the checksum message is transformed from quadrigation to summation. Therefore, the calculation of its probability density is the sum of several independent and equally distributed input information elements, like the variable message. The probability density can be calculated by a convolution, and the convolution operation can be transformed into a product operation in the Fourier transform domain. Figure 4.19 shows the transformation representation of a node-correction function verified by a Fourier transform in a bidirectional graph.

In an irregular Tanner graph, the degree-distribution sequences of the variable node and check node are $\{\lambda_1, \lambda_2, \dots, \lambda_{d_v}\}$ and $\{\rho_1, \rho_2, \dots, \rho_{d_c}\}$; then, the density-evolution expression of a variable message is

$$P^{(l+1)} = F^{-1} \left(F(P^{(0)}) \gamma \left(F(Q^{(l)})^{j-1} \right) \right), \quad (4.84)$$

where $Q^{(l)} = F^{-1}(\rho(F(P^{(l)})))$.

b. Parameterization of density evolution

The decoding threshold is determined by the channel parameters and LDPC code-construction parameters. Given code parameters $\gamma(x)$ and $\rho(x)$, the density function is the monotonic increment function of the channel characteristic parameter α , and there is an extreme value α' . When $\alpha \leq \alpha'$, the algorithm converges with a high probability [3, 40]. Extreme value α' is defined as the decoding threshold. Given bit rate R_c , the upper limit of channel-characteristic parameter α_c of channel capacity $C = R_c$ is the Shannon limit under this bit rate. The difference between α' and upper

limit α_c represents the distance between the optimal performance of the code set and Shannon's limit.

For example, for an AWGN channel with a zero mean value, we set $N_0 = 2\sigma_c^2$ as the unilateral noise power-spectral density, where σ_c is the upper limit parameter, the modulation amplitude is $x_0 \in \{+1, -1\}$, and Shannon's limit of the bit SNR is $E_b/N_0 = x_0^2/2R_c\sigma_c^2$. The distance between the decoding threshold α' and σ_c is defined as $\Delta_s = 20\log_{10}(\sigma_c/\sigma')$ dB.

Table 4.2 lists the confidence-propagation decoding threshold of several regular LDPC codes and their distance from Shannon's limit.

The analysis of a continuous-message density evolution is based on an acyclic tree graph. Richardson et al. proposed the clustering theorem that states that the performance of iterative algorithms on an acyclic graph asymptotically converges to an acyclic graph [11]. When the code length is sufficiently large, the statistical average performance of the decoder converges uniformly to α' .

For any channel parameter $\alpha \leq \alpha'$, if the code length is sufficiently large, when the number of iterations approaches infinity, any regular code can reliably transmit information. Correspondingly, for any error probability, after the corresponding number of iterations when the code length N tends to infinity, any regular code can exponentially approach the error probability depending on the code length, according to the probability 1.

Gallager pointed out that the necessary condition for constructing LDPC codes near capacity; i.e., the maximum checksum degree, is that j must approach infinity. In the study of Tornado codes, Luby et al. found that codewords constructed by irregular graphs had better performance [42], and clearly made the degree sequence close to the channel capacity of the binary deletion channel (BDC). Davey and Mackay [11] found that irregular LDPC codes with optimized degree sequences could effectively improve the decoding threshold and make the decoding performance closer to the channel capacity by applying the density evolution theory.

Because the performance of irregular LDPC codes is better than that of regular codes, the search for irregular LDPC codes with better degree-sequence distributions has become one of the research hotspots. Non-regular LDPC-code degree sequences are a nonlinear constraint-satisfaction problem in a multi-bit continuous space, and it is very difficult to design an optimal sequence that approximates the optimal degree sequence.

Table 4.2 Decoding thresholds in an AWGN channel and their distance from Shannon's limit

j	k	R_c	α'	α_c	E_b/N_0	$\Delta_s(\text{dB})$
3	6	0.5	0.88	0.979	0.184	0.926
4	8	0.5	0.83	0.979	0.184	1.434
5	10	0.5	0.79	0.979	0.184	1.863
3	5	0.4	1.00	1.148	-0.230	1.199
4	6	0.333	1.01	1.295	-0.484	2.159
3	4	0.25	1.26	1.549	-0.790	1.794

Richardson et al. used the hill-climbing method to obtain an optimized degree sequence that approached the channel capacity through a computer search. First, target error probability ε and a maximum number of iterations m were determined. Given a degree-sequence pair (γ, ρ) in advance, the LDPC code constructed by the degree-sequence pair (γ, ρ) was calculated by density evolution to meet the maximum-channel parameter value of ε . When (γ, ρ) was changed to $(\tilde{\gamma}, \tilde{\rho})$, if the LDPC code constructed by degree-sequence pair $(\tilde{\gamma}, \tilde{\rho})$ could obtain a larger channel-parameter value while satisfying ε , the LDPC code performance at this time was considered to be closer to the channel capacity.

Through the above methods, Richardson obtained the degree-sequence distribution of 1/2-rate LDPC codes in an AWGN channel as follows:

$$\begin{aligned} \gamma(x) = & 0.17120x + 0.21053x^2 + 0.00273x^3 + 0.00009x^6 + 0.15269x^7 + 0.09227x^8 \\ & + 0.02802x^9 + 0.12206x^{14} + 0.07212x^{29} + 0.25830x^{49} \end{aligned} \quad (4.85)$$

$$\rho(x) = 0.33620x^8 + 0.08883x^9 + 0.57497x^{10}. \quad (4.86)$$

The decoding threshold can reach $\sigma' = 0.9718$. Compared with the data in Table 4.2, this value is close to the capacity limit. It can be seen from the degree-sequence distribution that there is little difference in the degree of the check nodes, while there is a great difference in the degree of the variable nodes.

The density-evolution theory is a multi-parameter dynamic system that is very difficult to apply in complex channels. Hu [31], Echard [34] simplified the theoretical model of density evolution into a single-parameter-model dynamic system by studying the density evolution theory and applying the Gaussian approximation principle, to further simplify the analysis of the decoding threshold and convergence. The Gaussian approximation principle also provides the best or near-best quantization-decoding design basis for a given code.

4.5 LDPC-Code Construction

4.5.1 π -Rotating LDPC-Code Coding Principle

The performance of an LDPC code depends on the structure of the checksum matrix. A sparse random matrix is suitable for general performance analyses. However, if a field-programmable gate array (FPGA) or application-specific integrated circuit (ASIC) is used, the checksum matrix is better with a certain structure, which requires fewer hardware resources to store the matrix. The performance of an atmospheric laser-communication system and space optical-communication system before and after a π -rotating LDPC code is analyzed.

The random method is used to construct the sparse matrix for LDPC codes, and the parameter selection of the codeword is flexible. However, it is difficult to avoid four-line loops when constructing high-bit-rate, medium and short LDPC codes, and the constructed codes are not systematic or cyclic. Although the check matrix is sparse, the generation matrix obtained by the Gaussian elimination of the check matrix H during coding is not sparse. The encoder design is complicated and its complexity is $O(n^2)$. Rich Echard and Shih-Chun Chang proposed π -rotating LDPC codes [38, 39], which can be represented by a displacement vector, and the coding complexity has a linear relationship with the code length, which is easier to implement in hardware.

The check matrix H is decomposed into $H = [H^p \ H^d]$, where H^p is the upper triangular submatrix of order $M \times M$ in double-diagonal form, which is called the check-bit matrix. H^d is a matrix of order $M \times K$, called the information-bit matrix. For example, a 6×6 H^p is

$$H^p = I + D = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (4.87)$$

where I is the identity matrix, and D is I with the first row removed and an all-zero vector appended to the last row. Accordingly, the code vector C corresponding to matrix H is decomposed into $C = [C^p \ C^d]$. Therefore, the relationship between matrix H and the code vector is as follows:

$$HC^T = [H^p \ H^d] \begin{bmatrix} C^p \\ C^d \end{bmatrix} = H^p C^p + H^d C^d = 0. \quad (4.88)$$

From Eq. (4.88),

$$H^p C^p = v = H^d C^d \quad (4.89)$$

$$C^p = [H^p]^{-1} \cdot v = U^p \cdot v \mod (2), \quad (4.90)$$

where U^p is the upper triangular matrix and v is called the projection vector. The codeword is calculated using the information vector and check matrix H as follows:

1. Using Eq. (4.89), calculate v from C^d and H^d .
2. v and C^p are vectors with length M , and U^p is the upper triangular matrix. The process of calculating C^p from Eq. (4.90) is

$$\begin{aligned}
C^p(M) &= v(M) \\
C^p(i) &= v(i) + C^p(i+1) \quad (i = M-1, M-2, \dots, 1).
\end{aligned}
\tag{4.91}$$

4.5.2 π -Design of Rotation Matrices and Submatrices

In an m -order permutation matrix, each row and column have only one non-zero element (0, 1) matrix. Using m , a unit matrix is obtained by replacing any elements in a few rows (or columns). A π -rotation matrix is an m -order permutation matrix, with bits of the information matrix rotated by π minus H^d of the rotation matrix, according to certain rules. If H^d is composed of $q \times t$ π -rotation matrices, then H^d is a $qm \times tm$ -dimensional matrix with row weights t and column weights q . According to $H = [H^p \ H^d]$, the length of the information-bit vector C^d is tm , the code length N is $(q+t)m$, and the bit rate is $t/(q+t)$.

Matrix π_A can also be expressed as a permutation vector, in which each element represents the position of the non-zero elements in each column of the matrix π_A , in order from bottom to top. The permutation vector can be generated from the key vector of the index of π_A . The following describes how to generate the permutation vector from the index vector.

Define an index vector $[m, a, b]$ with three elements, m representing the length of the permutation vector, and a and b as integers.

Initialization: Set $S = \{s_0, s_1, \dots, s_{m-1}\}$, where $s_0 = 0, s_1 = 1, \dots, s_{m-1} = m-1$. The generation algorithm of the permutation vector is as follows.

```

    l ← m
    for j1 = 1 to m
        i ← a × j1 + b mod (m + 1 - j1)
        πA(j1) ← m - si
        for j2 = i to l - 1
            sj2 ← sj2+1
        next j2
        l ← l - 1
    next j1

```

For example, when the index is $[6, 4, 3]$ and $\pi_A = [5 \ 4 \ 1 \ 3 \ 2 \ 6]$, the corresponding matrix is

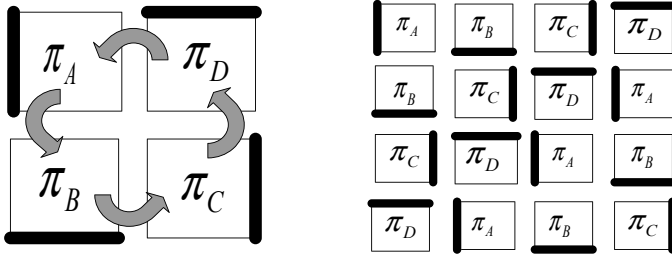


Fig. 4.20 Generation of four π -rotation matrices and the structure of the H^d matrix when the bit rate is $1/2$

$$\pi_A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \quad (4.92)$$

For the selection of the index vector, please refer to reference [40]. The π -rotation matrix is denoted as π_A ; it can be obtained by rotating 90° counterclockwise or clockwise. π_C and π_D can be obtained by successive rotations; hence, the name rotation matrix. When the bit rate is $1/2$, with matrix structure H^d , where $q = t = 4$, the dimension of the check matrix H is $4m \times 8m$, as shown in Fig. 4.20.

Figure 4.21 shows the performance comparison of the π -rotating LDPC code and Mackay randomly constructed LDPC code in an AWGN channel. The bit rate is $1/2$ and the information-bit lengths are similar. When the bit-error rate (BER) is 10^{-6} , the π -rotating LDPC code is slightly better than that of the random code. Figure 4.22 shows the short-code performance comparison when the bit rate is $1/2$ with a given index vector [41].

4.5.3 Extending Non-Regularity π -Rotate LDPC Code

The parity-check matrix of extended irregular π -rotated LDPC codes is shown in Fig. 4.23 [42]. The parity matrix consists of an information-bit matrix and a parity-bit matrix, and its dimension is $16m \times 20m$. The element represented by the diagonal is 1, and the blank part represents all 0 elements. Matrices A , B , C , and D are the compressed matrices of the original π_A , π_B , π_C , and π_D matrices.

Fig. 4.21 Performance comparison of rotation codes and random codes [14]

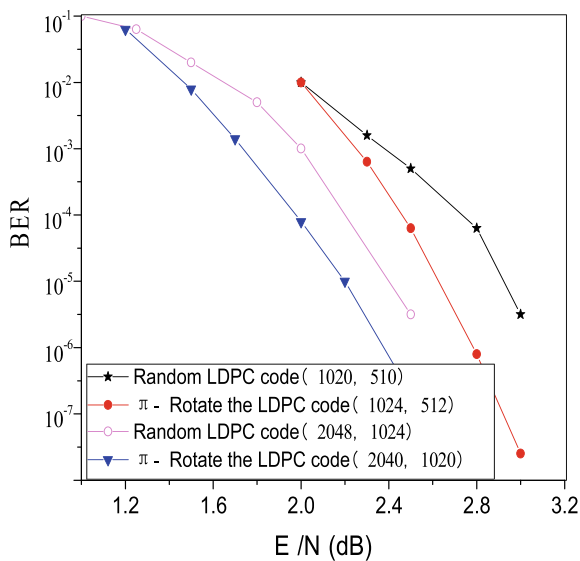
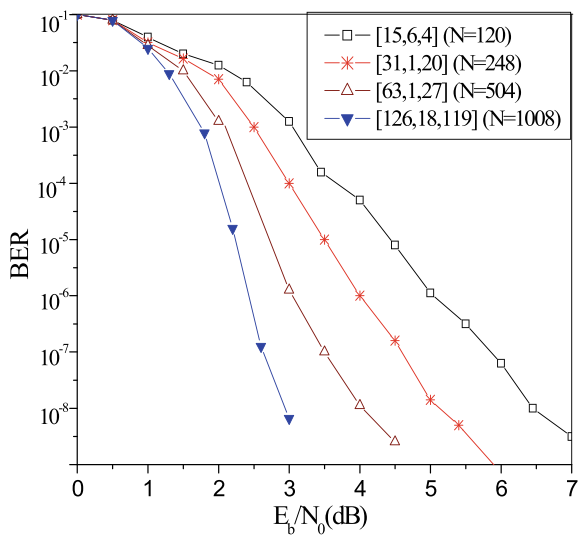
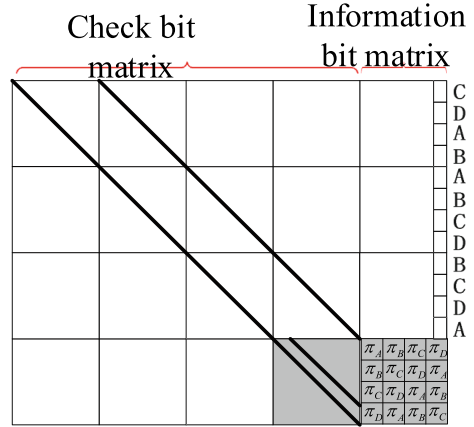


Fig. 4.22 Performance comparison of short codes at a bit rate of 1/2 [14]



$$A = \pi_A \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix}_{m \times (m/\alpha)}, \quad (4.93)$$

Fig. 4.23 Typical extended-irregular-code check matrix when the π -rotation matrix is $4m \times 8m$

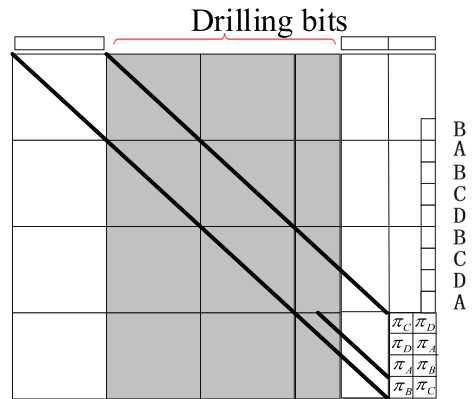


where I represents an $(m/\alpha) \times (m/\alpha)$ -dimensional identity matrix and A is an $m \times (m/\alpha)$ -dimensional matrix. In general, α must be divisible by m , which is called the compression factor. Matrices B , C , and D are obtained similarly.

Figure 4.24 shows the structure of an extended π -rotation-code check matrix. It can be seen that the typical π -rotation matrix can be defined by an index vector containing three parameters; however, the extended π -rotation matrix needs to be represented by four parameters, which are as follows:

- Inclusion of information segments I_{vec} : The number of displacement matrices in the columns of the information-bit matrix can be 1, 2, 3, or 4, and the corresponding information-bit lengths are m , $2m$, $3m$, or $4m$, as shown in Fig. 4.23 for $I_{vec} = 4$ and Fig. 4.24 for $I_{vec} = 2$.
- Puncturing of parity segments, P_{vec} . The code length can be shortened and flexible bit rates can be obtained by regularly deducting some elements; i.e., by punching holes in the parity-bit matrix. In Fig. 4.23, the parity-bit matrix from left to right

Fig. 4.24 Verification matrix when $I_{vec} = 2$, $P_{vec} = 10$, $H_{stack} = 9$



has a total of $16m$ columns, and is denoted as $c_1^{pe} \cdots c_{16}^{pe}$ in the unit of m . The punching starts at c_5^{pe} , with P_{vec} indicating the number of columns in m that the punched portion contains. The shaded part $c_5^{pe} \sim c_{14}^{pe}$ in Fig. 4.24 represents the punched bits; hence, $c_5^{pe} - c_{14}^{pe}$.

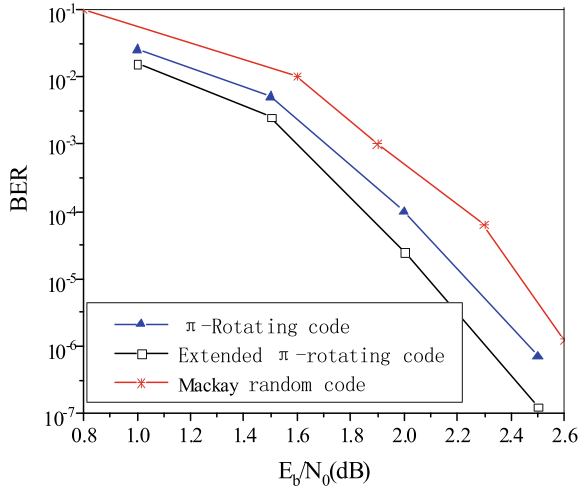
- H_{stack} indicates the number of compression matrices vertically stacked in the information-bit matrix. The value ranges from 0 to 12, as shown in Fig. 4.23 and $H_{stack} = 12$ in Fig. 4.24.
- The amount of compression used in the stack is α .

Extended irregular π -rotating LDPC codes can be represented by vectors $[m, a, b, I_{vec}, P_{vec}, H_{stack}, \alpha]$ with seven integer parameters. The bit rate of the extended irregular π -rotation code is defined as

$$R = \frac{I_{vec}}{16 + I_{vec} - P_{vec}}. \quad (4.94)$$

Therefore, the LDPC code rate shown in Fig. 4.24 is 1/4. Figure 4.25 shows the performance comparison of the Mackay random code, π -rotating LDPC code with parameters [126, 18, 119] and π -rotating LDPC code with parameters [126, 18, 119, 4, 12, 7, 3] in an AWGN channel with bit length 504 and bit rate 1/2 [42].

Fig. 4.25 Performance comparison of the Mackay random code, π -rotation code and extended code [14]



4.6 Performance Analysis of Euclidean Geometric LDPC Codes

To study the performance of Euclidean geometric LDPC codes in an optical pulse-position modulation (PPM) communication system, we simulate a link model of atmospheric optical communication. In the simulation, the atmospheric laser-communication system adopts an intensity modulation/direct detection (IM/DD) system, the modulation mode adopts on-off keying (OOK) and PPM modulation, the channels have lognormal-distributed, gamma-gamma-distributed, and K-distributed atmospheric turbulence, and the decoding method adopts the log-likelihood-ratio-based belief-propagation (LLR-BP) algorithm.

In the simulation, geometric LDPC codes can have arbitrary code lengths and code rates; therefore, the check matrix of the regular LDPC codes used in the simulation is generated by the Euclidean geometric construction method, with fixed column and row weights. Three regular Euclidean geometric LDPC codes with different code lengths and code rates are selected in the simulation. A Euclidean geometric LDPC code with a 0.686 code rate and 255-bit code length is constructed for analyzing and comparing the performance of the Euclidean geometric LDPC code with that without error-correction code in an atmospheric-turbulent optical-communication system.

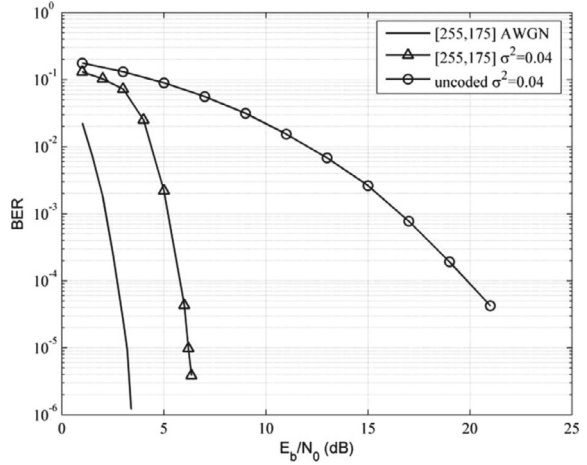
For the analysis and comparison of the influence of the maximum number of iterations on the bit-error performance, a Euclidean geometric LDPC code with a 0.822 bit rate and 4095-bit code length is constructed. To analyze the influence of PPM modulation on the bit-error performance, the applications of two-slot, four-slot, and 16-slot PPM modulation in three atmospheric-turbulent optical communication channels with different turbulence intensities are analyzed in detail.

4.6.1 Performance Comparison Between Uncorrected Codes and Corrected Codes

It is obvious that forward error-correcting codes can improve the reliability of an atmospheric laser-communication system. The error performance with and without forward error-correcting codes will be analyzed and compared below.

Figure 4.26 shows the first class of codewords with cyclic structures constructed using the Euclidean geometry method. Based on a GF(2) domain, when s is 4, EG-LDPC codes with a code length of 255, bit rate of 0.686, and row and column weights of 16 are constructed. To illustrate the influence of codewords on the system error performance in a turbulent channel, we use a log-domain BP decoding algorithm and set the maximum number of iterations to 20. We use [255, 175] codes to compare the BER performance in a log-normal-distributed turbulent channel with $\sigma^2=0.04$ using two-slot PPM modulation. We also analyze the BER performance in the same turbulent channel using only two-slot PPM modulation and demodulation.

Fig. 4.26 Performance comparison between an unencoded signal and an EG-LDPC [255, 175] encoding



It can be seen that the BER performance of the corrected codes decreases faster than that of the uncorrected codes. By observing the uncoded performance curve of $\sigma^2=0.04$ in Fig. 4.26 in a weakly turbulent channel, it can be seen that EG-LDPC code [255, 175] can obtain an encoding gain of approximately 14 dB when the BER is 10^{-4} . In an AWGN (Gaussian) channel, the EG-LDPC code [255, 175] is used to encode, and the simulation results using BPSK modulation indicate that when the BER is 10^{-5} in an AWGN channel, $\sigma^2=0.04$ the performance of the lognormal-distributed turbulent channel is improved by 3 dB.

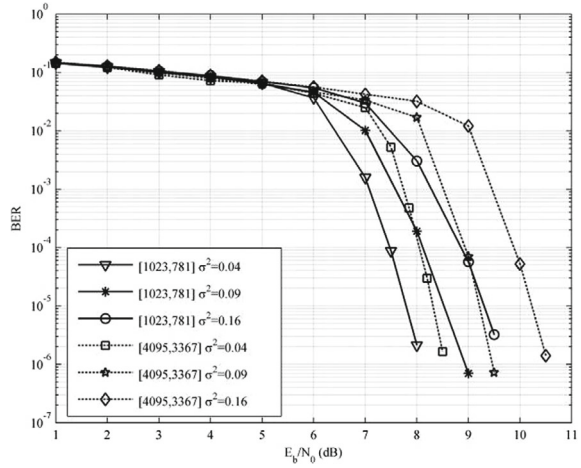
4.6.2 Performance Comparison of Euclidean-Geometric LDPC Codes with Different Bit Rates

Considering the Euclidean geometric LDPC codes introduced in Sect. 4.3, we analyzed and compared the performance of two codewords with the different code lengths and code rates in Table 4.1 when S is 5 and 6 respectively, as shown in Fig. 4.27.

To verify the soft iterative-decoding performance of EG-LDPC codes with different parameters in an atmospheric-turbulence channel, the BER performance of two groups of EG-LDPC codes with code lengths of 1023 and 4095 bits, as mentioned in Table 4.1, was simulated in a 2-PPM modulation channel with weak atmospheric turbulence, with turbulence intensities of 0.04, 0.09, and 0.16. The decoding algorithm adopted here is the soft-iteration algorithm discussed in the third section, and the maximum iteration number is set to 20.

As shown in Fig. 4.27, the BER performance curves of a [1023, 781] EG-LDPC code with a 0.763 bit rate and row weight of 32 and a [4095, 3367] EG-LDPC code with a 0.822 bit rate and row weight of 64 can be seen under different turbulence

Fig. 4.27 Performance of EG-LDPC codes with different parameters under different atmospheric-turbulence intensities



intensities. The BER performance of both decreases gradually with the increase of σ^2 . Because the former has a higher bit rate (0.763) than the latter (0.822), it has better performance under the same parameters; however, the information-transmission efficiency is lower. In addition, it can be seen that the waterfall performance of the former is lower than that of the latter, because the latter has a higher line weight and the minimum distance of the code is also larger.

4.6.3 Performance Comparison of LDPC Codes in Different Turbulent Channels

In atmospheric laser communication, the signal can easily be affected by atmospheric turbulence, which leads to serious degradation of the beam quality, a further decrease of the bit-error rate and channel capacity of a free-space optical (FSO) communication system, and can seriously affect the stability and reliability of the communication system. Below, we analyze and compare the effects of turbulent channels with different intensities on the bit-error performance in detail.

In Fig. 4.28, the first type of codeword with a cyclic structure that is constructed using Euclidean geometry is based on a GF(2) field. When S is 5, an EG-LDPC [1023, 781] with code length of 1023, code rate of 0.763, and row and column weights of 32 is encoded. OOK, 2, 4, and 16-PPM modulation were adopted, and the BP decoding algorithm was selected. When the number of iterations was 20, they were respectively in the lognormal-distribution turbulent channel with $\sigma^2 = 0.04$ (as shown in Fig. 4.28b) and the gamma-gamma-distribution turbulent channel with $\alpha=4.0$, $\beta=1.9$ (as shown in Fig. 4.28c). The bit-error rate was analyzed in a K -distributed turbulent channel with $K = 2$ (as shown in Fig. 4.28d).

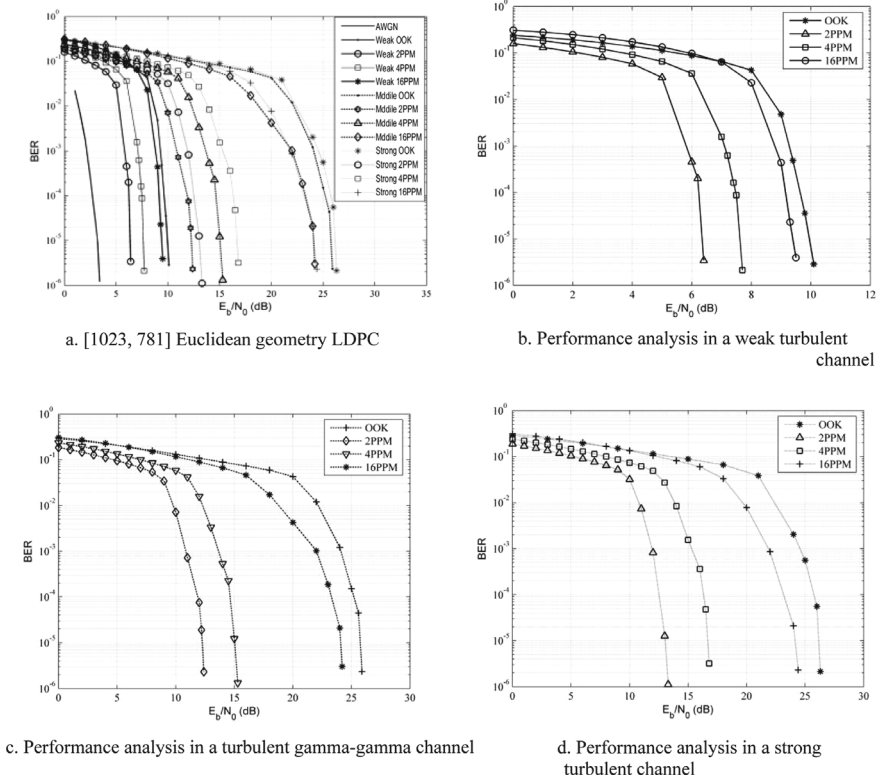


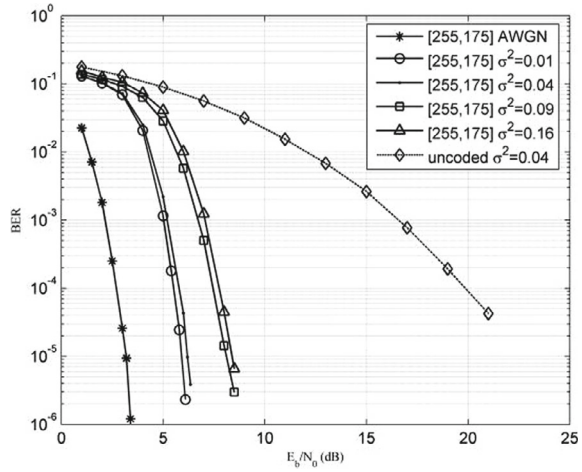
Fig. 4.28 Performance analysis of EG-LDPC [1023, 781] codes in different turbulent channels

As can be seen from the comparison in the figure, when the modulation mode is constant, the BER performance is significantly reduced with the increase of turbulence intensity. For the same turbulence intensity, different modulation modes also affect the BER performance. As can be seen from Fig. 4.28, under certain BER conditions, 2-PPM has the best performance, while OOK has the worst modulation performance. It can be seen that the turbulence intensity and modulation mode greatly influence the channel bit-error performance.

For EG-LDPC codes with a code length of 255, bit rate of 0.686, and row and column weights of 16, a log-domain BP decoding algorithm is adopted, with the maximum iteration number set to 20. In an AWGN channel with turbulence intensity σ^2 , the simulation results of the BER performance in a 2-PPM modulation channel with weak atmospheric-turbulence intensities of 0.01, 0.04, 0.09, and 0.16 are shown in Fig. 4.29.

As can be seen from the performance curves in Fig. 4.29, compared with the decoding performance of the EG-LDPC [255, 175] code in an AWGN channel, the EG-LDPC code's decoding performance deteriorates gradually with the increase

Fig. 4.29 Bit-error rate performance of EG-LDPC [255, 175] codes in a lognormal turbulent channel



of turbulence intensity σ^2 in a turbulent-atmospheric channel. The result is consistent with the actual physical effect, because the introduction of turbulent noise will inevitably deteriorate the error-correction performance of the decoder. The greater the turbulent-noise intensity, the greater the possibility of error, and thus, the higher the bit-error rate. By comparing the uncoded performance curve with $\sigma^2 = 0.04$ in Fig. 4.29 in the weakly turbulent channel, it can be seen that the EG-LDPC code [255, 175] can obtain a coding gain of about 14 dB when the BER is 10^{-4} .

4.6.4 Performance Comparison with RS Codes

In Fig. 4.30, EG-LDPC [255, 175] codes with a cyclic structure are constructed based on the GF(2) domain. When S is 4, the code length is 255, the code rate is 0.686, and the row and column weights are 16. In a lognormal-distributed turbulent channel where $\sigma^2 = 0.04$ and a gamma-gamma-distributed turbulent channel where $\alpha = 4.0$, $\beta = 1.9$, the simulation results are compared with RS [255, 223] codes, when the number of iterations is 20, using two-slot PPM modulation and the LLR-BP decoding algorithm.

In Fig. 4.30, AWGN is the performance curve of the EG-LDPC [255, 175] code when BPSK modulation is used in an AWGN channel. In a weakly turbulent channel, when the BER is 10^{-5} , the error performance of the EG-LDPC codes is about 8 dB higher than that of the RS codes. In a medium turbulent channel, when the BER is 10^{-5} , the error performance of the EG-LDPC codes is about 20 dB higher than that of the RS codes. It can be seen that the error performance of the EG-LDPC codes is obviously better than that of the RS codes.

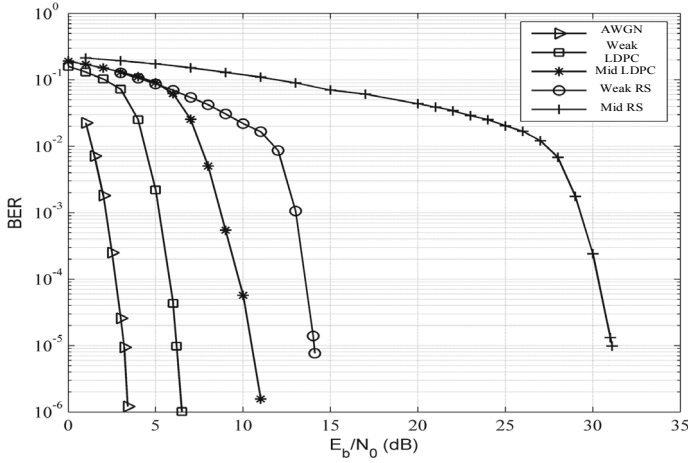


Fig. 4.30 Performance analysis of EG-LDPC [255, 175] codes and RS [255, 223] codes in different turbulent channels

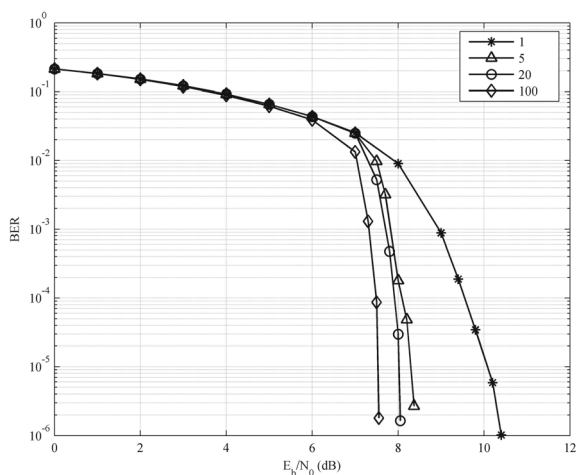
4.6.5 Performance Comparison of Different Numbers of Iterations

When the LLR-BP decoding algorithm is adopted, the number of decoding iterations will have a certain impact on the decoding result. Following is the analysis and comparison of the error performance under different numbers of iterations.

Because EG-LDPC codes generally have high row and column weights, they can be decoded with fewer iterations in practical applications; thus, reducing the processing delay of the decoder. To verify the decoding performance of an EG-LDPC code in an atmospheric-turbulence channel with different iterations, Monte Carlo simulation was used to examine the decoding performance of the EG-LDPC [4095, 3367] code shown in Table 4.1 at different maximum iterations. The results are shown in Fig. 4.31. Here, the turbulence intensity of the channel $\sigma^2=0.04$.

As shown in Fig. 4.31, the BER performance curves when the maximum number of iterations is 1, 5, 20, and 100 shows that the decoding performance deteriorates gradually as the number of iterations decreases. When the maximum number of iterations is 5, the performance gain is only about 0.8 dB less than when the BER is 10^{-5} ; however, the decoding complexity and delay are lower.

Fig. 4.31 Performance of EG-LDPC [4095, 3367] code with different maximum iterations in an atmospheric-turbulent channel



4.7 Hardware Implementation of an LDPC Code Encoder

4.7.1 General Block Diagram of the Encoder Implementation

An LDPC code takes a known information bit, calculates the check bit, and then forms the codeword. The encoder processes the encoding in frames. The encoder inputs a continuous flow of information bits. Through the grouping module, an information-bit flow can be divided into K bits as a group; through the string and transformation module, the K bits of information can be in parallel. Then, they are input into the coding module, which generates M check bits, and attaches them to an information bit to form an N -encoding codeword. With the π -rotating LDPC-code coding principle, a pipeline-processing method can be realized. Figure 4.32 shows the overall structure of the LDPC-code encoder hardware.

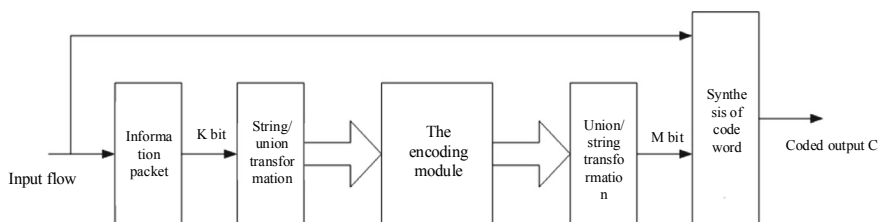


Fig. 4.32 Block diagram of the encoder

4.7.2 Coding Implementation of π -Rotating LDPC Code

The steps to generate codewords from the information vector and check matrix H are as follows.

Calculate v from C^d and H^d .

$$\begin{aligned} v &= H^d C^d \\ H^p C^p &= v \\ C^p &= [H^p]^{-1} \cdot v = U^p \cdot v \pmod{2}, \end{aligned} \quad (4.95)$$

where v and C^p are column vectors of dimension M , and U^p are upper triangular matrices. Therefore, the calculation of C^p starts from the bottom of v , and the first bit at the bottom is the last bit of C^p . The calculation process is as follows:

$$\begin{aligned} C^p(M) &= v(M) \\ C^p(i) &= v(i) + C^p(i+1) \quad (i = M-1, M-2, \dots, 1). \end{aligned} \quad (4.96)$$

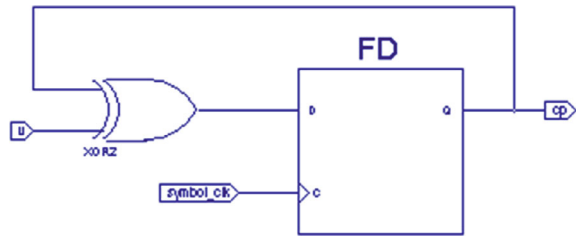
v and C^p can be calculated using the circuit shown in Fig. 4.33.

The matrix H^d shown in Fig. 4.33 is composed of four columns. The information bits are divided into four subvectors of length m , which are, respectively, c_1^d, c_2^d, c_3^d and c_4^d , as shown in Eq. (4.97). Thus, four subvectors, v_1, v_2, v_3 , and v_4 , of length m of v can be obtained.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} \pi_A & \pi_B & \pi_C & \pi_D \\ \pi_B & \pi_C & \pi_D & \pi_A \\ \pi_C & \pi_D & \pi_A & \pi_B \\ \pi_D & \pi_A & \pi_B & \pi_C \end{bmatrix} \begin{bmatrix} c_1^d \\ c_2^d \\ c_3^d \\ c_4^d \end{bmatrix} \quad (4.97)$$

It can be seen from Eq. (4.97) that four inner-product operations are required for each subvector in the matrix operation of v_1, v_2, v_3 , and v_4 . We take v_1 as an example. When permutation matrix $\pi_A = [5 \ 4 \ 1 \ 3 \ 2 \ 6]$, the calculation system $\pi_A c_1^d$ is shown in Fig. 4.34. c_1^d is input to the lower register, and $\pi_A c_1^d$ is output to the left register. In Fig. 4.35, $\pi_A c_1^d$ is assigned from the left register to a similar register

Fig. 4.33 Circuit realization of v and C^p calculation



above; c_2^d is input, $\pi_B c_2^d$ is computed, and $\pi_B c_2^d$ is output to the upper register, while $\pi_A c_1^d + \pi_B c_2^d$ is computed. The process is repeated. Inputs c_3^d and c_4^d are rotated and all the inner-product results are summed to obtain v_1 .

Similarly, we can compute subvectors v_2 , v_3 , and v_4 to obtain v . According to Fig. 4.35, c^p can be obtained from v to form a codeword and complete the coding.

Fig. 4.34 First inner-product calculation
 $\pi_A c_1^d$

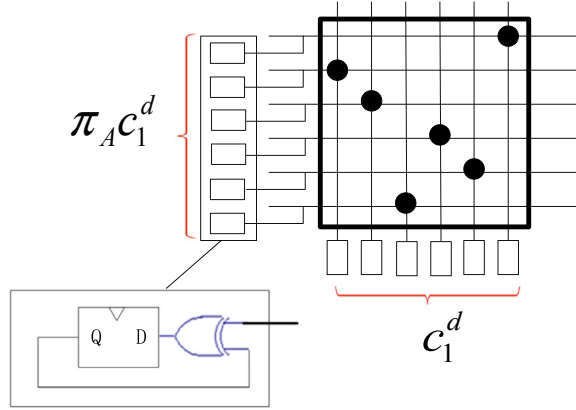
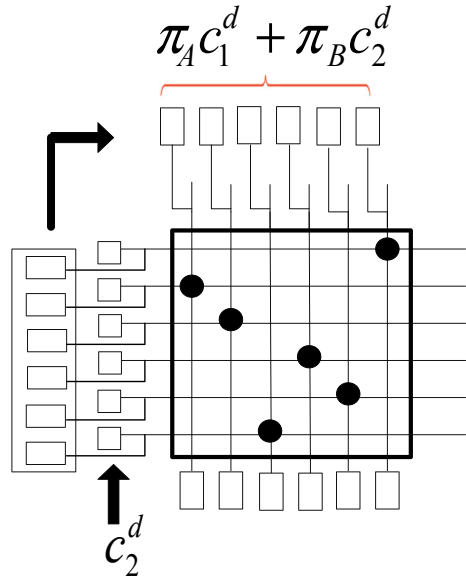


Fig. 4.35 Second inner-product calculation
 $\pi_A c_1^d + \pi_B c_2^d$



4.7.3 Pipeline Processing of the Coding

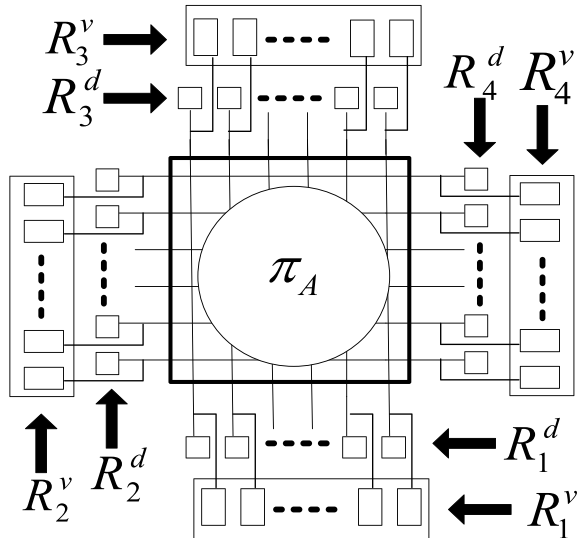
Π -rotating LDPC-code coding has potential high-speed coding characteristics. Each sub-vector of the information vector rotates around the permutation matrix, and all the sub-vectors of v can be obtained; thus, the coding can be processed using a pipeline. Figure 4.36 shows the encoding-implementation diagram, wherein R_1^v, R_2^v, R_3^v and R_4^v are called the mapped memory.

All registers are initialized. When c_1^d is entered into register R_1^d , the product $\pi_A c_1^d$ is stored in R_2^v ; subvector c_1^d is then rotated from R_1^d to R_2^d , and product $\pi_B c_1^d$ is entered into R_3^v . Similarly, by rotating c_1^d , the resulting products $\pi_C c_1^d$ and $\pi_D c_1^d$ are stored in R_4^v and R_1^v , respectively. c_1^d proceeds through a cycle of rotation calculation, and then is output through the buffer.

Before c_2^d is input, the value stored in the mapping memory is rotated around π_A by 1/4; that is, R_1^v is assigned to R_2^v , R_2^v to R_3^v , R_3^v to R_4^v , and R_4^v to R_1^v . Note that data cannot be overwritten in the assignment process. c_2^d enters through R_1^d to perform the same operation as c_1^d . c_2^d passes through a cycle of rotation calculation, and is then output through the buffer. At this point, the values in the mapping register are

$$\begin{aligned} R_1^v &= \pi_C c_1^d + \pi_D c_2^d \\ R_2^v &= \pi_D c_1^d + \pi_A c_2^d \\ R_3^v &= \pi_A c_1^d + \pi_B c_2^d \\ R_4^v &= \pi_B c_1^d + \pi_C c_2^d. \end{aligned} \quad (4.98)$$

Fig. 4.36 Coding implementation diagram [14]



We proceed to rotate the value stored in the mapping store 1/4 of the way around π_A , and then enter a third information subvector c_3^d . After the rotation calculation, the values in the mapped register are

$$\begin{aligned} R_1^v &= \pi_B c_1^d + \pi_C c_2^d + \pi_D c_3^d \\ R_2^v &= \pi_C c_1^d + \pi_D c_2^d + \pi_A c_3^d \\ R_3^v &= \pi_D c_1^d + \pi_A c_2^d + \pi_B c_3^d \\ R_4^v &= \pi_A c_1^d + \pi_B c_2^d + \pi_C c_3^d. \end{aligned} \quad (4.99)$$

The value stored in the mapping memory is rotated further 1/4 times around π_A , and a fourth information subvector c_4^d is entered. After a rotation calculation, the values in the mapped register are

$$\begin{aligned} R_1^v &= \pi_A c_1^d + \pi_B c_2^d + \pi_C c_3^d + \pi_D c_4^d \\ R_2^v &= \pi_B c_1^d + \pi_C c_2^d + \pi_D c_3^d + \pi_A c_4^d \\ R_3^v &= \pi_C c_1^d + \pi_D c_2^d + \pi_A c_3^d + \pi_B c_4^d \\ R_4^v &= \pi_D c_1^d + \pi_A c_2^d + \pi_B c_3^d + \pi_C c_4^d. \end{aligned} \quad (4.100)$$

Mapping sub-vectors v_1, v_2, v_3 and v_4 are stored in the respective mapping registers. Check vector c^p is calculated using the device shown in Fig. 4.35 to complete the coding.

4.7.4 Coded ModelSim Simulation

In the above coding process, we can choose

$$\pi_A = [2 \ 7 \ 12 \ 21 \ 10 \ 8 \ 25 \ 13 \ 19 \ 3 \ 29 \ 1 \ 26 \ 14 \ 30 \\ 20 \ 18 \ 4 \ 5 \ 15 \ 27 \ 9 \ 22 \ 11 \ 16 \ 28 \ 23 \ 24 \ 17 \ 6].$$

According to the verification-matrix structure in Fig. 4.33, $m = 30$, the bit rate is $1/2$, and the code length is 240. The Verilog hardware description language (HDL) is used, Xilinx Company's comprehensive ISE software is adopted, and the Xilinx-Spartan II-series XC2S200 FPGA chip is used. The simulation in Fig. 4.37a shows the computation of $\pi_A c_1^d$, based on the ModelSim 6.0 software, in which CLK is the system clock, C1 is the input information subvector, and V1A is the value of computing $\pi_A c_1^d$.

Figure 4.37b shows the ModelSim simulation diagram that realizes Eq. (4.97). When the input of 120-bit information vector CD is known, the output mapping vector is V. Figure 4.37c shows the simulation diagram of the input information-bit CD and output codeword C, where CP is the verification code calculated by CD.



Fig. 4.37 ModelSim simulation diagrams implemented using coding [14]

4.8 LDPC-Code Decoding Implementation

4.8.1 General Block Diagram of the Decoder

For a semi-randomly constructed π -rotating LDPC code, the logarithmic-domain BP decoding algorithm is adopted, so a large number of multiplication operations are not needed. In fact, only Eq. (4.62) has multiplication operations, and it can be realized using an XOR operation. Figure 4.38 shows the overall decoding structure.

4.8.2 Data-Input Module

The input to the decoder is always a constant stream, bit by bit, frame by frame. The π -rotation LDPC code has a code length of 240 and a bit rate of 1/2. The 120-bit check bit is only related to the 120-bit information bits. Therefore, the 240-bit codes are processed as a whole during decoding, and then the codeword, composed of the next 240 received codes, is decoded. The 240 codewords are called a frame of data.

The data-input module contains two memories, RAM1 and RAM2, a serial-parallel conversion circuit, and synchronous detection. In the decoding process, data

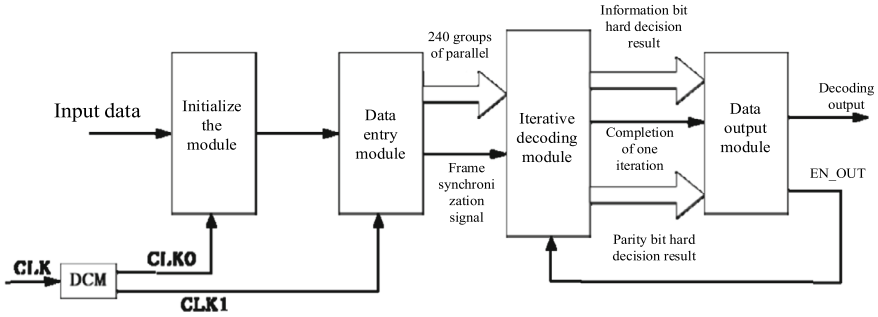


Fig. 4.38 Overall decoding structure [14]

are entered into the decoder in serial mode and stored in memory through a serial–parallel transformation. An incoming frame is stored in memory RAM1 and is read by a variable-node unit (VNU) at each iteration.

By detecting the frame-synchronization signal, the next frame is determined to be input to the input module, and the memory module is set to cache the data of the second frame, until the data of this frame is fully entered into RAM2. When frame 2 is fully in memory, it begins the iterative decoding process. Simultaneously, the first frame data-decoding iteration ends, and the memory is used to store the input data of the third frame. Therefore, the initialization module needs to store two frames of data simultaneously, one for the current frame and one for the next frame.

4.8.3 Parallel Iterative-Decoding Module

LDPC-code decoding is based on a binary graph composed of variable nodes, parity nodes, and edges. The update information of the variable node is only provided by the check node through the edge, and then the variable node calculates the update information to be transmitted to the check node according to the received update information. The verification node only needs to provide the information needed by the variable node through the edge, and then the verification node calculates the information to be transmitted to the variable node according to the update information received.

During the decoding design, each variable node can be assigned a variable-node processing unit (VNU) and each check node is assigned a processing unit (CNU). The iteration module is the core of the decoder, which calculates the variable-node and parity-node information, exchanges data, and updates.

In the design, a cache RAM can be set between the VNU and CNU to avoid the difficult-to-control and unstable iteration state brought by a direct connection between the VNU and CNU; this is beneficial to synchronization, as shown in

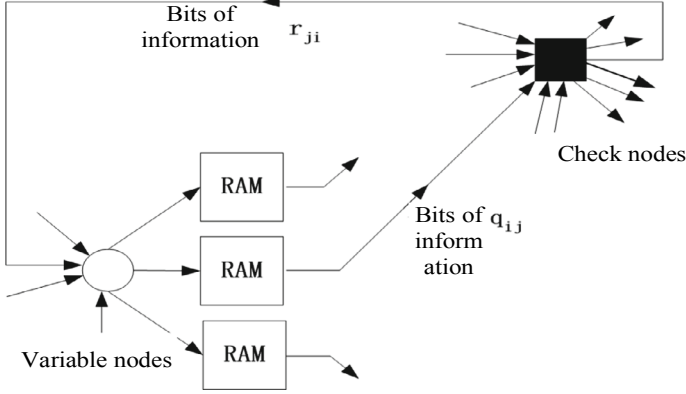


Fig. 4.39 Parallel-decoding data path [14]

Fig. 4.39. We only draw the connection mode of one data path between two node-processing units; the connection modes of the data paths between the other node-processing units are the same. After the channel-initialization information and data are sent by the CNU to the VNU for information processing, the data are input to the CNU for a data update through the cache RAM; finally, the data are fed back to the VNU, completing an iteration.

For regular-code decoding, only a certain number of VNUs, CNUs, and RAM in between need to be copied. However, for irregular codes, the numbers of VNUs and CNUs are not equal; hence, the hardware implementation is very complicated.

We observe the π -rotation LDPC code in Fig. 4.40. When $m = 30$, the degree of variable node x_1 is 1, the degree of variable nodes $x_2 - x_{120}$ is 2, and the degree of variable nodes $x_{121} - x_{240}$ is 4. The degree of check node $z_1 - z_{119}$ is 6, and the degree of check node z_{120} is 5. In this manner, we can also simplify decoding by copying the RAM between the VNU and CNU.

(1) Variable-node processing unit (VNU)

Equation (4.67) can be written as $L(q_{ij}) = L(P_i) + \sum_{j \in C(i)} L(r_{ji}) - L(r_{ij})$, and (4.68) can be written as $L(Q_i) = L(P_i) + \sum_{j \in C(i)} L(r_{ji})$. Assuming that the degree of variable node i is d_v , the processing unit of the variable node should have d_v data-input and -output channels. Figure 4.40 is the schematic diagram.

(2) Verifying the node-processing unit (CNU)

The degree of the verification node j is d_c , and the verification-node processing function (4.66) is written as

$$L(r_{ij}) = \text{sign}(\alpha_{ij}) \cdot \prod_{i \in R(j)} \text{sign}(\alpha_{ij}) \cdot f^{-1} \left\{ \left(\sum_{i \in R(j)} f(\beta_{ij}) \right) - f(\beta_{ij}) \right\}. \quad (4.101)$$

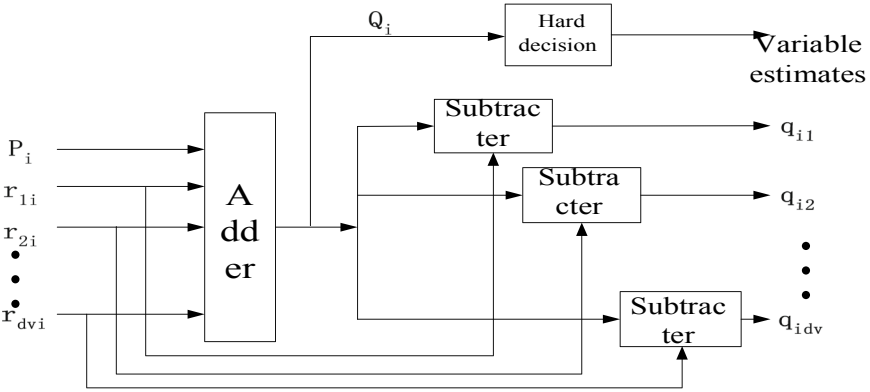


Fig. 4.40 Implementation structure of a variable node unit [14]

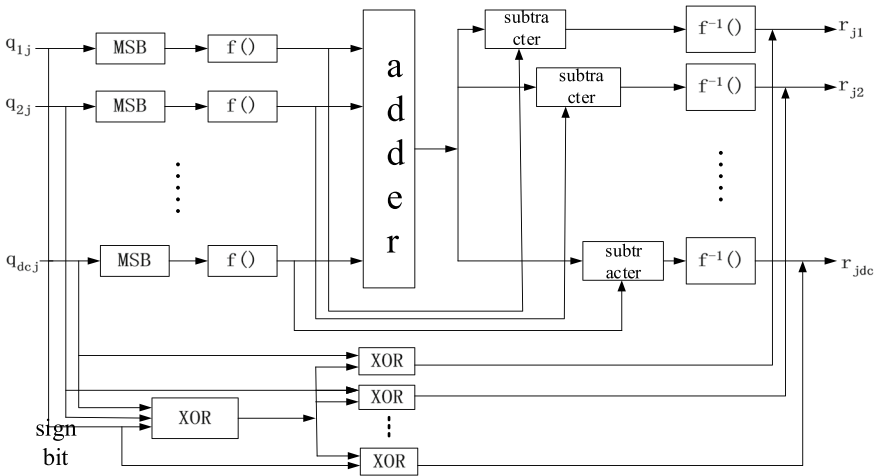


Fig. 4.41 Implementation structure of the verification node [14]

In this manner, Eq. (4.101) only needs XOR and addition and subtraction operations to complete. $f(x)$ can be realized by establishing a look-up table, as shown in Fig. 4.41. The Most Significant Bit (MSB) module masks the highest bit.

4.8.4 Data-Output Module

The data-output module judges the success of each decoding, and controls the iteration times and the parallel-series conversion of the output data. If the decoding succeeds or the maximum number of iterations is reached, the output module informs

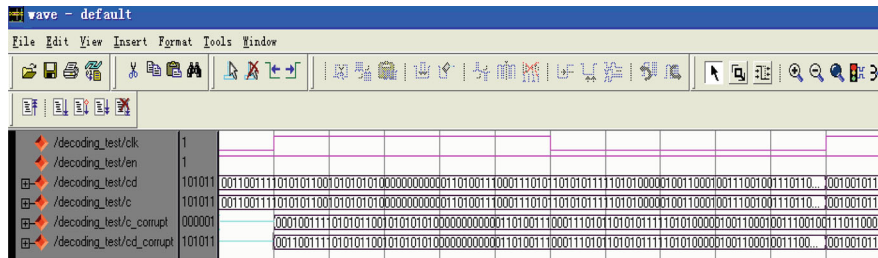


Fig. 4.42 System simulation waveform [14]

the decoding module to stop the current decoding by raising the En_out signal, and then reads the next frame of data for decoding. If the decoding succeeds, the serial data are output.

The electronic-design automation (EDA) software mainly used in the design of the LDPCs includes ModelSim and ISE. The Verilog HDL and schematic-diagram input method are adopted in the design-input, design-compilation, functional-simulation, and design-modification parts in the design process. In addition, the generated Verilog code is simulated in ModelSim and ISE.

Figure 4.42 shows the ModelSim simulation waveform of the system; CLK represents the simulation clock, EN represents the system-enable signal, high-level effective, CD represents the system-input information, C represents the code stream after coding, and C_corrupt represents the signal output due to channel interference; that is, the signal input by the decoder. Cd_corrupt indicates the signal output after decoding. Errors occurred in the third and 50th bits of the C_corrupt input side of the decoding, which were corrected after simulation.

References

1. Ke XZ, Xi XL (2004) Introduction to wireless laser communication. Beijing University of Posts and Telecommunications Press, Beijing, pp 12–16

2. Wang XM, Xiao GZ (2001) Error correcting code—principle and method. Xidian University Press, Xi ‘an, pp 34–55

3. Wen H, Fu Y, Zhou L (2006) Principle and application of LDPC code. University of Electronic Science and Technology of China Press, Chengdu, pp 24–38

4. Gallager RG (1962) Low density parity check codes. IEEE Trans Inform Theory 8(3):208–220

5. Gallager R (2008) Low-density parity-check codes. J Circ Syst 8(1):3–26

6. MacKay DJC (1996) Near Shannon limit performances of low density parity check codes. Elect Lett 32(8):1645–1646

7. Tanner RM (1981) A Recursive approach to low complexity codes. IEEE Trans Inform Theory 27(5): 533–547

8. Zhong HM, Wang RH (2004) Error correction coding technology LDPC code in 4G and its new progress. Guangdong Commun Technol 12:6–8

9. Richrdson TJ, Shokrollahi MA (2001) Design of capacity approaching irregular low-density parity-check codes. IEEE Trans Inform Theory 47(2):619–637

10. Luby MG, Mitzenmacher M (1998) Analysis of low density codes and improved designs using irregular graphs. In: Proceedings of the 30th ACM symposium on theory of computing, pp 249–258
11. Richardson TJ, Urbanke RL (2001) The capacity of low-density parity-check codes under message-passing decoding. *IEEE Trans Inform Theory* 47(2):599–618
12. Davey MC, Mackay DJC. Monte Carlo simulations of infinite low density parity check codes over GF(q). <http://wol.ra.phy.can.ac.uk/is/papers>
13. Davey MC, MacKay DJC (1998) Low density parity check codes GF(q). *IEEE Commun Lett* 2(6):165–167
14. Jia KJ (2007) Design and implementation of LDPC code in atmospheric laser communication system. Xi'an University of Technology, Xi'an
15. Davey MC, Mackay DJC. Monte Carlo simulations of infinite low density parity check codes over GF(q). <http://wol.ra.phy.can.ac.uk/is/papers>
16. Davey MC (1999) Error-correction using low-density parity-check codes. University of Cambridge 3(9):123–157
17. MacKay DJC (1999) Good error-correcting codes based on very sparse matrices. *IEEE Trans Inf Theory* 45(2):399–431
18. MacKay DJC, Wilson ST, Davey MC (1999) Comparison of constructions of irregular Gallager codes. *IEEE Trans Commun* 46:1449–1454
19. Richardson TJ, Urbanke RL (2001) Efficient encoding of low-density parity-check codes. *IEEE Trans Inform Theory* 47(2):638–656
20. Karkooti M (2004) Semi-parallel architectures for real-time LDPC coding. Rice University, Houston, pp 13–18
21. Wang P, Wang XM (2004) Fast coding of LDPC codes. *J Xidian Univ* 31(6):934–938
22. Zhou Q, Zhang HB, Pan Y (2006) Improvement of greedy algorithm in LDPC coding. *Telecommun Technol* 4:68–72
23. Pirou F (2004) Introduction of Low-density Parity-Check decoding algorithm design. Master thesis in Electronics Systems at Linköping Institute of Technology, Sweden, pp 13–18
24. Yuan DF, Zhang HX (2006) Advanced Channel coding technology for broadband mobile communication, no 3. Beijing University of Posts and Telecommunications Press, Beijing, pp 20–25
25. Berrou C, Glavieux A, Thitimajshima P (1993) Near Shannon limit error-correcting coding and decoding: turbo-codes. In: Proceedings of IEEE international communications conference, pp 1064–1070
26. Chung SY, Richardson TJ, Urbanke RL (2001) Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation. *IEEE Trans Inform Theory* 47(2):657–670
27. Sun SH (2002) Research on several key problems of low density checksum code. Xidian University, Xi'an, pp 22–25
28. Chung SY, Richardson TJ (2001) On the design of low-density parity-check codes within 0.0045 dB of Shannon limit. *IEEE Commun Lett* 5(2):1–3
29. Chung SY (2000) On the construction of some capacity-approaching coding schemes. Massachusetts Institute of Technology, pp 13–16
30. He YC (2002) Low density check code theory and its application based on graph model]. Xidian University, Xi'an, pp 11–14
31. Luby MG, Mitzenmacher M, Shokrollahi MA, etc. Stemann (1997) Practical loss-resilient codes. In: Proceeding of 29th annual symposium theory computing, pp 150–159
32. Hu XY, Eleftheriou E, Arnold DM (2001) Progressive edge-growth Tanner graphs. In: Globecom 2001—IEEE global telecommunications conference, no 1, November 2001, pp 980–984
33. Marc PC, Fossorier, Mihaljevic M et al (1999) Reduced complexing iterative decoding of low-density parity check codes based on belief propagation. *IEEE Trans Commun* 47(5):673–680
34. Wang HX (2001) On permutation matrix. *J Shenyang Petrochemical Coll* 17(3):54–57
35. Echard R (2002) On the construction of some deterministic low-density parity-check codes. George Mason Univ Fairfax, VA, pp 13–15

36. Echard R, Chang SC (2005) Design considerations leading to the development of good π -rotation LDPC codes. *IEEE Commun Lett* 9(5):233–236
37. Echard R, Chang SC (2003) The extended irregular π -rotation LDPC codes. *IEEE Commun Lett* 7(5):230–233
38. Fan L, Wang L (2006) FPGA implementation of irregular LDPC decoder for medium and long frames. *Electron Eng* 32(8):21–24
39. Chang XM (2003) Verilog—HDL practice and application system design. Beihaung University Press, Beijing, pp 132–137
40. Hu HF, Zhang JX (2006) Performance and analysis of LDPC code in PPM channel of space optical communication. *Space Electron* 3:20–25
41. Hu HF (2005) Application of LDPC code in space optical communication. Zhejiang University of Technology, Hangzhou, pp 22–27
42. Tong S (2004) Low density check code theory and quantization decoding. Xidian University, Xi'an, pp 15–17
43. Wang C, Xue XG, Zhong XC (2004) FPGA/CPLD design tool—Xilinx ISE application in detail. Posts and Telecommunications Press, Beijing, pp 132–145

Chapter 5

Research on Polar Codes in Optical Wireless Communication Systems



Using channel polarization, the polar codes considers the mutual information between the input and output when encoding, so that it can use the symmetric capacity of the channel, while ensuring the error-correction performance. In terms of coding and decoding, the polar-code complexity is almost linear with the length of the codeword, which is low. The error rate decreases rapidly with the increase in codeword length, which is significant, considering the strict requirements for the bit-error rate in deep-space communications and optical transmission systems.

5.1 Polar Codes in Optical-Wireless Channels

5.1.1 Polar-Code Coding

The polar code is similar to the RM (Reed–Muller) code and belongs to the coset-code category. For a polar code whose code length is, $N = 2^n n \geq 0$, u_1^N represents the input sequence, x_1^N represents the coding sequence, and G_N represents the generation matrix; then, the polar code is

$$x_1^N = u_1^N G_N \quad (5.1)$$

Assuming that A is any subset of the set $\{1, \dots, N\}$, because polar codes are linear code, Eq. (5.1) can be rewritten as

$$x_1^N = u_A G_N(A) \oplus u_{A^c} G_N(A^c) \quad (5.2)$$

where $G_N(A)$ represents a matrix composed of rows indexed in set A in G_N , A^c represents the complement of set A in set $\{1, \dots, N\}$, and u_A is a subset of the input sequence u , which is determined by the elements in set A . u_{A^c} is a “frozen bit”;

generally, it can be considered to be composed of all zeroes. Therefore, the polar code can be represented by a G_N -coset code with parameters $(N, K, \mathcal{A}, u_{\mathcal{A}^c})$, where K is the size of set \mathcal{A} , and K/N is the code rate. In this manner, polar-code coding is simplified to determining set \mathcal{A} and generating matrix G_N .

The polar-code encoding process is mainly divided into the following three steps:

1. Determine the information-bit set \mathcal{A} . According to different channel models, select appropriate methods to calculate the symmetric capacity $I(W)$ or Bhattacharyya parameter $Z(W)$ of the channel. Select the channel with the larger $I(W)$ or smaller $Z(W)$ to transmit information. The remaining channel is used to transmit frozen-bit information. Because frozen bits only play a role in decoding, they can be considered to be composed of all zeros without losing generality.
2. The generating matrix G_N is generated, and according to the determined information-bit set \mathcal{A} , the coding matrix $G_N(K)$ is determined to be used for coding. K is the size of information-bit set \mathcal{A} and matrix $G_N(K)$ is composed of rows corresponding to information-bit set \mathcal{A} in matrix G_N .
3. Coding. Assuming that u_1^K Represents the Input Information and x_1^N Represents the Encoded Information, the Polar Code is Encoded According to Eq. (5.2).

(1) Generating matrix G_N

Let the binary representation of i be $(b_1 \cdots b_n)_2$, and the binary expansion of j be $(b'_1 \cdots b'_n)_2$, where $b_i, b'_i \in \{0, 1\}$; then, the (i, j) term in matrix G_N is

$$(G_N)_{i,j} = \prod_{i=1}^n (1 \oplus b'_i \oplus b_{n-i} b'_i) \quad (5.3)$$

where $n = \log N$. If the encoder is implemented using matrix multiplication, according to Eq. (5.3), the asymptotic complexity of the encoding is $O(N^2)$. Note that in Eq. (5.9), assuming that the density of one element in $F^{\otimes n}$ is D_n , Kronecker's power exponent is defined as

$$F^{\otimes n} = \begin{bmatrix} F^{\otimes n-1} & 0 \\ F^{\otimes n-1} & F^{\otimes n-1} \end{bmatrix} \quad (5.4)$$

The following recursive relation about D_n can be obtained through observation:

$$D_n = \frac{3}{4} D_{n-1}, \quad (5.5)$$

where $D_1 = 3/4$. Therefore, $D_n = (3/4)^n$; thus, D_n decreases rapidly with the increase of n . Therefore, the polar-code generation matrix is sparse and can be encoded with lower complexity.

The generation matrix G_N of a polar code is essentially a linear-transformation matrix, assuming that, for all $n \geq 0, N = 2^n$. Using I_k to represent the k -dimensional identity matrix, the recursive definition of generation matrix G_N is

$$G_N = (I_{N/2} \otimes F)R_N(I_2 \otimes G_{N/2}), \quad N \geq 2, \quad (5.6)$$

where $G_1 = [1]$.

Because $(I_{N/2} \otimes F)R_N = R_N(F \otimes I_{N/2})$, the above formula can be rewritten as

$$\begin{aligned} G_N &= R_N(F \otimes I_{N/2})(I_2 \otimes G_{N/2}) \\ &= R_N(F \otimes G_{N/2}). \end{aligned} \quad (5.7)$$

$G_{N/2} = R_{N/2}(F \otimes G_{N/4})$ in the above formula is replaced by $G_{N/2}$. The mixed product of the Kronecker product has the following property:

$$\begin{aligned} B_N &\triangleq R_N(I_2 \otimes R_{N/2})(I_4 \otimes R_{N/4}) \cdots (I_{N/2} \otimes R_2) \\ (AC) \otimes (BD) &= (A \otimes B)(C \otimes D). \end{aligned} \quad (5.8)$$

Therefore, let $A = I_2$, $B = R_{N/2}$, $C = F$, and $D = F \otimes G_{N/4}$; then,

$$\begin{aligned} G_N &= R_N(F \otimes (R_{N/2}(F \otimes G_{N/4}))) \\ &= R_N(I_2 \otimes R_{N/2})(F^{\otimes 2} \otimes G_{N/4}). \end{aligned} \quad (5.9)$$

We repeat the above operations to obtain the final generation matrix G_N ; that is,

$$G_N = B_N F^{\otimes n}. \quad (5.10)$$

where; then, its recursive form is

$$B_N = R_N(I_2 \otimes B_{N/2}). \quad (5.11)$$

R_N is a “reverse shuffle” operation, which is actually a circular left shift, based on the binary-element sequence number. For example, there are s_1^N vectors ($s_1^N \triangleq (s_1, s_2, \dots, s_N)$), and $v_1^N = s_1^N R_N$; then, $v_{b_1 b_2 \dots b_n} = s_{b_2 \dots b_n b_1}$, for all $b_1, b_2, \dots, b_n \in \{0, 1\}$. Among them, $v_{b_1 b_2 \dots b_n}$ and $s_{b_2 b_3 \dots b_1}$ are the binary representations of vectors v and s , respectively, where $b_1 b_2 \dots b_n$ and $b_2 \dots b_n b_1$ are elements. Therefore, matrix B_N can be regarded as a bit-reversal operation.

For example, Fig. 5.1 shows the specific implementation of a bit-inversion operation when $N = 8$.

As can be seen from Fig. 5.1, the first row $\{000, 001, 010, 011, 100, 101, 110, 111\}$ is the binary representation of the natural sequence $\{0, 1, 2, 3, 4, 5, 6, 7\}$ of matrix column numbers. We rotate each column number one bit to the left. The first four digits of the generated new sequence are even numbers $\{000, 010, 100, 110\}$, and the last four digits are odd numbers $\{001, 011, 101, 111\}$. It can be observed that these two parts are natural sequences. Then, rotate left again, and finally obtain a series of numbers $\{0, 4, 2, 6, 1, 5, 3, 7\}$. Therefore, the entire operation B_8 is a bit-reversal operation. The final result

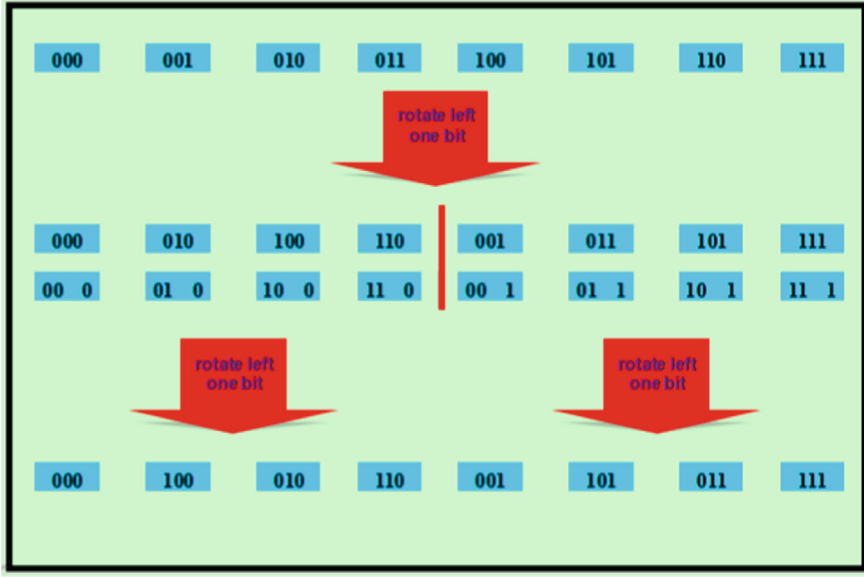


Fig. 5.1 Specific implementation of a bit-inversion operation, where $N = 8$

of this operation is that the output is given in a natural sequence after using successive-cancellation (SC) decoding.

(2) Information-bit set \mathcal{A}

According to the polar-code coding principle, channel polarization occurs after channel merging and channel separation. Some of the channels have a symmetric capacity close to 1, and another part close to 0. Information is transmitted on the channels with a symmetric capacity close to 1; the part close to 0 is used to transmit fixed frozen bits. Therefore, the essential problem of polar-code coding is to select the appropriate information-bit set \mathcal{A} , according to the channel polarization.

Theoretically, determining information-bit set \mathcal{A} involves calculating the symmetric capacity $I(W_N^{(i)})$ of the channel. However, for non-binary-erasure channels (non-BECs), it is very difficult to calculate $I(W_N^{(i)})$, or even approach an accurate calculation value. When the symmetric capacity $I(W_N^{(i)})$ is close to 1, the Bhattacharyya reliability parameter $Z(W_N^{(i)})$ is close to 0. Therefore, given a channel W with code length N , after the channel combination and separation, a group of channels $\{W_N^{(i)} : 1 \leq i \leq N\}$ is obtained. The task of polar-code construction is to select the group of channels with the highest reliability, that is, determine set \mathcal{A} . Polar-code construction is equivalent to optimizing the following problem.

$$\begin{aligned} \min_{\mathcal{A} \subset \{1, \dots, N\}} \sum_{i \in \mathcal{A}} Z(W_N^{(i)}) \\ \text{s.t. } |\mathcal{A}| = K. \end{aligned} \quad (5.12)$$

If the Bhattacharyya parameter $Z(W_N^{(i)})$ of each channel can be calculated, we select the smallest K of the $Z(W_N^{(i)})$ values (k is the dimension of the code) through a sorting algorithm; that is, information-bit set \mathcal{A} is determined. At present, the Bhattacharyya parameter $Z(W_N^{(i)})$ is mainly calculated using the following three methods:

a. Recursive construction

This construction method was proposed for BEC channels. Because the Bhattacharyya parameter of a BEC channel is equal to the deletion probability, if the deletion probability of BEC channel W is ε , then, the Bhattacharyya parameter $Z(W_N^{(i)})$ of channel $W_N^{(i)}$, obtained after the channel-polarization operation can be calculated using the following recursive method:

$$Z(W_N^{(2j-1)}) = 2Z(W_{N/2}^{(j)}) - Z(W_{N/2}^{(j)})^2 \quad (5.13)$$

$$Z(W_N^{(2j)}) = Z(W_{N/2}^{(j)})^2. \quad (5.14)$$

The recursion termination condition is $Z(W_1^{(1)}) = \varepsilon$. Using this recursive approach, the asymptotic algorithm complexity is only $O(N)$.

This method is not applicable for general channels; however, a similar construction algorithm is proposed in reference [1]. Assuming that the symmetric capacity of channel W' is $I(W')$, for BEC channels with a deletion probability of $1 - I(W')$, the above recursive construction is used to calculate the value of the Bhattacharyya parameter $Z(W_N^{(i)})$, and this calculation result is used as the construction result of channel W' .

Although this method is not optimal because it does not consider the characteristics of channel W' , the polar codes constructed using this method usually have quite good performance. This method is very simple to implement, and the complexity of the algorithm is low.

b. Monte-Carlo simulation construction

For more general channels, Arikan proposed a Monte-Carlo simulation method to estimate the Bhattacharyya parameter $Z(W)$ [2]:

$$Z(W_N^{(i)}) = E \left[\sqrt{\frac{W_N^{(i)}(y_1^N, u_1^{i-1} | u_i \oplus 1)}{W_N^{(i)}(y_1^N, u_1^{i-1} | u_i)}} \right]. \quad (5.15)$$

We define random variable $R_N^{(i)}$ as

$$R_N^{(i)} \triangleq \frac{W_N^{(i)}(y_1^N, u_1^{i-1} | u_i \oplus 1)}{W_N^{(i)}(y_1^N, u_1^{i-1} | u_i)}. \quad (5.16)$$

Then,

$$Z(W_N^{(i)}) = E \left[\sqrt{R_N^{(i)}} \right]. \quad (5.17)$$

Note that $R_N^{(i)}$ is actually the likelihood ratio of u_i . For a given u_1^N and y_1^N , it can be calculated directly through the SC decoding method mentioned below. According to the given input distribution and channel-transition probability, multiple groups of u_1^N and y_1^N are generated. The SC decoding algorithm is used to calculate multiple groups of different $\sqrt{R_N^{(i)}}$ values, and the expectation can be estimated by taking their arithmetic average. The estimated value of $Z(W_N^{(i)})$ is obtained. Although the desired results can be obtained using this method, its computational complexity is high and it is difficult to realize.

c. Density evolutionary structure

The Bhattacharyya parameter $Z(W_N^{(i)})$ of the channel is the upper bound of the error probability of the channel $W_N^{(i)}$ under maximum-likelihood (ML) decoding. The error probability $Pe(W_N^{(i)})$ of channel $W_N^{(i)}$ under ML decoding can be calculated using the density evolution method. $Pe(W_N^{(i)})$ can also be used to determine the set \mathcal{A} ; thus, completing the construction of the polar code.

Although the polar codes for different channels can be constructed approximately using the above methods, there are no corresponding construction methods to accurately construct polar codes, except for the BEC channel.

(3) Example of polar-code coding

The polar-code coding process through a BEC channel is explained in detail. As polar and RM codes are both G_N -coset codes, this code can be represented by parameters $N, K, \mathcal{A}, u_{\mathcal{A}^c}$, where N is the code length, K is the size of set \mathcal{A} , and $u_{\mathcal{A}^c}$ represents the frozen bit. We calculate the Bhattacharyya parameter $Z(W)$ of the BEC channel when the code length is $N = 8$, as shown in Table 5.1.

According to the information-bit selection rules, assuming that a bit channel with Bhattacharyya parameter $Z(W_N^{(i)}) < 0.5$ is used for information transmission, the

$$\begin{aligned}
& \oplus [0 \ 1 \ 0 \ 1] \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
&= [0 \ 1 \ 0 \ 1 \ 0 \ 1 \ 0 \ 1] \oplus [0 \ 1 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0] \\
&= [0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1] \tag{5.20}
\end{aligned}$$

It can be seen from the coding process that selecting different information-bit sets \mathcal{A} will directly affect the final coding results; thus, selecting information-bit sets is a difficult point in polar-code coding. The essence of determining an information-bit set is identifying the polarization of a specific channel, which is the focus of polar-code theory at present.

The difference of the frozen bit also affects the coding result, and the frozen bit plays a vital role in decoding. In the SC decoding algorithm, the value is finally decoded according to the information sent and received using the known frozen bit. The information sent by the frozen bit is known only to the transmitting and receiving ends. Thus, even if the information is intercepted, to obtain the correct decoding information, both the information-bit set \mathcal{A} and the frozen-bit information must be known, which ensures the security of the information transmission. Thus, polar codes are used for cases with high-confidentiality requirements.

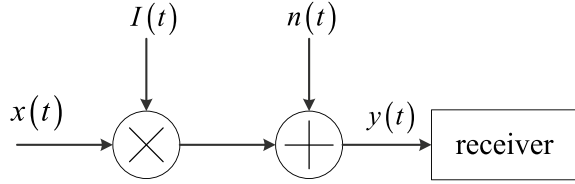
5.1.2 Optical-Wireless Channel Model

When an optical signal is transmitted in an atmospheric channel, it will be disturbed by atmospheric attenuation, free-space loss, atmospheric turbulence, and back-ground radiation, all of which affect the reliability and stability of optical wireless communication. Among these, the influence of atmospheric turbulence is the most prominent.

In terms of phenomena, atmospheric turbulence is manifested in random fluctuations (flicker), and the bending, expansion, distortion, and degradation of the spatial coherence of a laser beam in a certain section [3]. In an intensity modulation / direct detection (IM / DD) system, the intensity fluctuation, i.e., light-intensity flicker, is the most important factor affecting the performance of the communication system, and the flicker will increase the bit-error rate. According to the different atmospheric-structure coefficients C_n^2 , atmospheric turbulence can be divided into three categories [4]:

$$\begin{cases} C_n^2 \geq 10^{-12} \text{m}^{-\frac{2}{3}} & \text{strong turbulence} \\ C_n^2 \approx 10^{-14} \text{m}^{-\frac{2}{3}} & \text{medium turbulence} \\ C_n^2 \leq 10^{-16} \text{m}^{-\frac{2}{3}} & \text{weak turbulence} \end{cases} \tag{5.21}$$

Fig. 5.2 Equivalent mathematical model of an atmospheric channel



An atmospheric-turbulent optical communication channel is generally considered as a discrete-time channel with a time-varying intensity gain and additive Gaussian white noise (AGWN). When an optical signal is transmitted in an atmospheric channel, it will inevitably be affected by atmospheric attenuation and turbulence. The block diagram of the equivalent mathematical model of an atmospheric channel is shown in Fig. 5.2.

In Fig. 5.2, $x(t)$ shows the transmitted signal, $y(t)$ is the received signal, and $I(t)$ and $n(t)$ respectively represent the multiplicative and additive noises of the atmospheric channel; $I(t)$ and $n(t)$ are independent of each other. Then, the mathematical model of an atmospheric-turbulence channel can be described as

$$y(t) = I(t) \cdot x(t) + n(t). \quad (5.22)$$

The multiplicative noise $I(t)$ is the channel-state information, which represents the intensity of the atmospheric turbulence.

(1) Weak-Turbulence Channel

For a weak-turbulence channel, the light intensity follows a lognormal distribution, and its probability-density function (PDF) is [5]

$$f(I) = \frac{1}{I\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{(\ln I + \sigma_0^2/2)^2}{2\sigma_0^2}\right), \quad (5.23)$$

where σ_0^2 represents the variance of the lognormal distribution, which depends on the channel characteristics and is expressed as

$$\sigma_0^2 = \exp\left[\frac{0.49\delta^2}{(1 + 0.18d^2 + 0.56\delta^{12/5})^{7/6}} + \frac{0.51\delta^2}{(1 + 0.9d^2 + 0.62d^2\delta^{12/5})^{5/6}}\right] - 1, \quad (5.24)$$

where $d = \sqrt{kD^2/4L}$, $k = 2\pi/\lambda$ is the light wave number, $\delta^2 = 1.23C_n^2 k^{7/6} L^{11/6}$ is the Rytov variance, and C_n^2 is the atmospheric refractive-index structure constant, which is determined by the altitude. The peripheral model is usually the Hufnagel-Valley model, which is described as

$$C_n^2(h) = 0.00594(v/27)^2(10^{-5}h)^{10}e^{h/1000} + 2.7 \times 10^{-6}e^{-h/1500} + Ae^{-h/1000}, \quad (5.25)$$

where h is the altitude (m), v is the wind speed (m/s), and A is the value of $C_n^2(0)$. For a free-space optical (FSO) communication link near the ground, in the process of changing from weak to strong turbulence, the value of C_n^2 changes from $10^{-17}m^{-2/3}$ to $10^{-13}m^{-2/3}$, and generally takes $10^{-15}m^{-2/3}$ as the mean value; without a clear explanation, $\sigma_0 = 0.2$

(2) Medium-Turbulence Channel

In a turbulent channel in the atmosphere, the light intensity I follows the gamma-gamma distribution, and its probability-density function is [6]

$$f(I) = \frac{2(\alpha\beta)^{(\alpha+\beta)/2}}{\Gamma(\alpha)\Gamma(\beta)} I^{(\alpha+\beta)/2-1} K_{\alpha-\beta}(2\sqrt{\alpha\beta}I), \quad (5.26)$$

where $K_n(\cdot)$ represents the second type of modified Bessel function with order n and $\Gamma(\cdot)$ represents the gamma function. In the case of a plane wave, the scintillation-index parameters α and β are expressed as

$$\alpha = \frac{1}{\sigma_x^2} = \left[\exp\left(\frac{0.49\sigma_0^2}{(1 + 1.11\sigma_0^{12/5})^{7/6}}\right) - 1 \right]^{-1} \quad (5.27)$$

$$\beta = \frac{1}{\sigma_y^2} = \left[\exp\left(\frac{0.51\sigma_0^2}{(1 + 0.69\sigma_0^{12/5})^{5/6}}\right) - 1 \right]^{-1}. \quad (5.28)$$

Here, $\sigma_0^2 = 1.23C_n^2\kappa^{7/6}L^{11/6}$ is the Rytov variance. In the case of medium-strong turbulence, the scintillation index is related to α and β , that is:

$$\sigma_I^2 = \alpha^{-1} + \beta^{-1} + (\alpha\beta)^{-1}. \quad (5.29)$$

(3) Strong Turbulent Channel

In a strong atmospheric-turbulence channel, the light intensity follows the K-distribution, and its probability-density function is [7]

$$f(I) = \frac{2a^{(a+1)/2}}{\Gamma(a)} \cdot I^{(a-1)/2} K_{a-1}(2\sqrt{aI}), \quad (5.30)$$

where $\Gamma(\cdot)$ is the gamma function, K_n represents the second type of modified Bessel function with order n , and a is the channel parameter related to the effective number of

discrete scatterers. In the strong-turbulence range, the turbulence intensity decreases with the increase of a .

5.1.3 Polarization of an Atmospheric-Turbulence Channel

The idea of polar-code coding is to select the channel with a better symmetric capacity to transmit information among a group of channels that become polarized, while the channel with poorer symmetric capacity transmits the frozen bits. The symmetric capacity of channels other than the BEC channel cannot be accurately calculated, which greatly limits the application of polar codes. Therefore, how to accurately construct polar codes in non-BEC channels has become a difficult and hot topic in this field.

The channel used in optical wireless communication systems is an atmospheric channel, which is a complex non-BEC channel. At present, there is no corresponding construction method to accurately construct a polar code under an atmospheric channel. Through research on other non-BEC channels, such as AGWN and Rayleigh channels, it was found that, although polar codes cannot be constructed by accurately calculating the symmetric capacity of these channels, the polar codes constructed by some approximate methods also have excellent performance.

(1) Gaussian-Approximation Construction Method

The Bhattacharyya parameter $Z(W)$ of the channel is the upper bound of the error probability of channel W under maximum-likelihood decoding. The error probability $Pe(W)$ of channel W under maximum-likelihood decoding can be calculated using the density-evolution method. With this error probability, the information-bit set \mathcal{A} can also be determined, to construct and encode the polar code.

A Gaussian approximation is an approximate density-evolution method. Its basic idea is to analyze the infinite-dimensional message density in density evolution as following a Gaussian or Gaussian-mixture distribution. In a linear design, this can not only simplify the problem, but also greatly improve the accuracy of the approximation [8–10].

For a weak atmospheric-turbulence channel with light-intensity modulation, assuming that the channel-state information is a known fixed value within the transmission time interval of each information bit, the channel model is equivalent to a memoryless Gaussian channel with a binary input and continuous output, whose fading coefficient obeys a lognormal distribution [11]. At this time, the Gaussian-approximation (GA) method can be used to realize the polarization in a weak atmospheric-turbulence channel.

First, the definition of the channel Bhattacharyya parameter should be extended from a discrete channel to a continuous channel. For a continuous channel, the Bhattacharyya parameter is defined as

$$Z(W) \triangleq \int \sqrt{P(y|0)P(y|1)} dy, \quad (5.31)$$

where $P(y|0)$ and $P(y|z = 1) = \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{(y-I)^2}{N_0}\right)$, represent the probability of receiving y when sending “0” and “1” respectively.

For the weakly turbulent channel case, using on-off keying (OOK) modulation when sending “0” and “1”, the probability-density functions of the conditional distributions $P(y|x = 0)$ and $P(y|x = 1)$ are, respectively,

$$P(y|x = 0) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{y^2}{2\sigma^2}\right). \quad (5.32)$$

$$P(y|x = 1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-I)^2}{2\sigma^2}\right). \quad (5.33)$$

Therefore, the Bhattacharyya parameter $Z(W)$ of the weakly turbulent channel is

$$\begin{aligned} Z(W) &= \int_{-\infty}^{\infty} \sqrt{p(y|0)p(y|1)} dy \\ &= \int_{-\infty}^{\infty} \sqrt{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-I)^2}{2\sigma^2}}} dy. \\ &= e^{-\frac{I^2}{8\sigma^2}} \end{aligned} \quad (5.34)$$

To address the problem of effectively calculating the Bhattacharyya parameter of virtual channel $W_N^{(i)}$ in Eq. (5.34) for a weak atmospheric-turbulence channel W after the channel merging and splitting operations, the simplified calculation process using the GA method is given below.

For the weak atmospheric-turbulence channel model in Eq. (5.22), assuming that the transmitted and received channel-state information is known and I is a constant value in each symbol bit time, the log-likelihood ratio (LLR) of the output signal Y of channel W approximately obeys the Gaussian distribution with a mean value of $2/\sigma^2$ and variance of $4/\sigma^2$, where σ^2 is the variance of the AGWN of channel W .

We use $L_N^{(i)}$ to represent the output log-likelihood ratio $\log((p(y_i|0))/(p(y_i|1)))$ of the i th split channel $W_N^{(i)}$, and use $m_N^{(i)}$ to represent the mean value of $L_N^{(i)}$ corresponding to the i th bit; then, $m_N^{(i)}$ can be approximately calculated using the following recursive formula:

$$m_N^{(2i-1)} = f^{-1}\left(1 - \left(1 - f\left(m_{N/2}^{(i)}\right)\right)^2\right). \quad (5.35)$$

$$m_N^{(2i)} = 2m_{N/2}^{(i)}. \quad (5.36)$$

The calculation expression of function $f(x)$ is [12]

$$f(x) = \begin{cases} e^{\alpha x^\gamma + \beta} & x \leq 10 \\ \frac{1}{2} \left(\sqrt{\frac{\pi}{x}} e^{-\frac{x}{4}} \left(1 - \frac{3}{x}\right) + \sqrt{\frac{\pi}{x}} e^{-\frac{x}{4}} \left(1 + \frac{1}{7x}\right) \right) & x > 10 \end{cases}, \quad (5.37)$$

where $\alpha = -0.4527$, $\beta = 0.0218$, and $\gamma = 0.86$.

From the initial $m_1^{(1)} = 2/\sigma^2$, according to Eqs. (5.35) and (5.36), the mean value of all log-likelihood ratios $m_N^{(i)}$ of virtual channel $W_N^{(i)}$ can be calculated recursively. We use $(\sigma_N^{(i)})^2$ to represent the noise variance of the i^{th} virtual channel $W_N^{(i)}$. From the relationship between the variance and mean, the expression of $(\sigma_N^{(i)})^2$ can be obtained as

$$(\sigma_N^{(i)})^2 = 2/m_N^{(i)}. \quad (5.38)$$

We substitute Eq. (5.38) into (5.34) to obtain the Bhattacharyya parameters of $W_N^{(i)}$ in a weak turbulence channel:

$$Z(W_N^{(i)}) = e^{-I^2/(8(\sigma_N^{(i)})^2)} = e^{-I^2 m_N^{(i)}/16}. \quad (5.39)$$

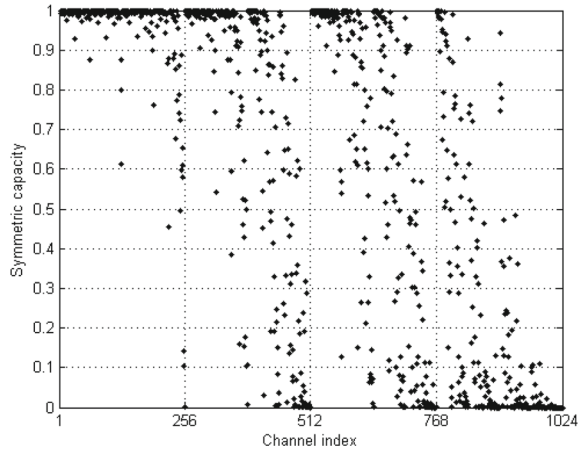
From this, we can calculate the Bhattacharyya parameter values of the virtual channel after merging and splitting, when the weak atmospheric-turbulence channel adopts the standard polarization-kernel matrix.

Figure 5.3 shows the distribution diagram of the Bhattacharyya parameters of the virtual channel, calculated according to Eq. (5.39) with code length $N = 1024$ and code rate $R = 0.5$. It shows the weak atmospheric-turbulence channel model at $\sigma_0 = 0.2$, where the horizontal axis is the index number i of the virtual channel and the vertical axis is the Bhattacharyya parameter $Z(W_N^{(i)})$, constrained by the symmetric capacity of the channel.

Because the Bhattacharyya parameter $Z(W_N^{(i)})$ and the channel symmetric capacity $I_N^{(i)}(W)$, under the discrete memoryless channel of any binary input satisfy a certain constraint relationship, it can be seen from the distribution in Fig. 5.3 that the GA method can be used for polar codes constructed for weak atmospheric-turbulent channels. In addition, the channel symmetry capacity $I_N^{(i)}(W)$ produces a polarization phenomenon where some tend to "0" and some tend to "1."

However, this Gaussian-approximation construction method can only be applied to a weak turbulence channel, because the noise of the weak turbulence channel obeys a lognormal distribution, which is approximately equivalent to a Gaussian channel. In medium and strong turbulence channels, owing to the influence of noise on the light intensity, the channel state at this time does not meet this approximately equivalent condition; thus, it cannot be processed like a weak turbulence channel.

Fig. 5.3 Polarization of a weak atmospheric-turbulence channel at $\sigma_0 = 0.2$ with a code length of $N = 1024$



(2) Recursive construction method

For constructing polar codes in non-BEC channels, Arikan proposed the Monte-Carlo construction method; however, this method has high computational complexity and is not easy to implement [13]. Then, he proposed a heuristic construction method, which can process other channels equivalent to BEC channels with the same channel capacity.

Assuming that the symmetric capacity of channel W' is $I(W')$, the value of Bhattacharyya parameter $Z(W_N^{(i)})$ is calculated by the recursive construction of Eqs. (5.13) and (5.14) for the BEC channel with a deletion probability of $1 - I(W')$; then, the result of this calculation is used as the construction result of channel W' . Although this method is not optimal, because it does not consider the characteristics of channel W' and the influence of noise, the polar codes constructed by this method usually have quite good performance. This method is very simple to implement and the algorithm complexity is low.

Zhao et al. used three different algorithms to encode and decode the polar code for the Bhattacharyya parameter $Z(W)$ in the literature [14], and gave a specific simulation analysis. The simulation results showed that the method of selecting information bits under a binary-deletion channel was better for binary-deletion channels, binary-symmetric channels, and binary-input AGWN channels.

Generally, this recursive method is used to construct polar codes, and the deletion probability is 0.5; that is, $Z(W_1^{(1)}) = 0.5$. Here, polar codes with a code length of 1024 are simulated in medium, strong, and weak turbulence channels, as shown in Fig. 5.4. It can be seen from the simulation results that the performance of the polar code recursively constructed using a BEC channel in a turbulent channel is also quite good. However, this construction method is not optimal in turbulent channels, because it does not consider the influence of the turbulent multiplicative noise and

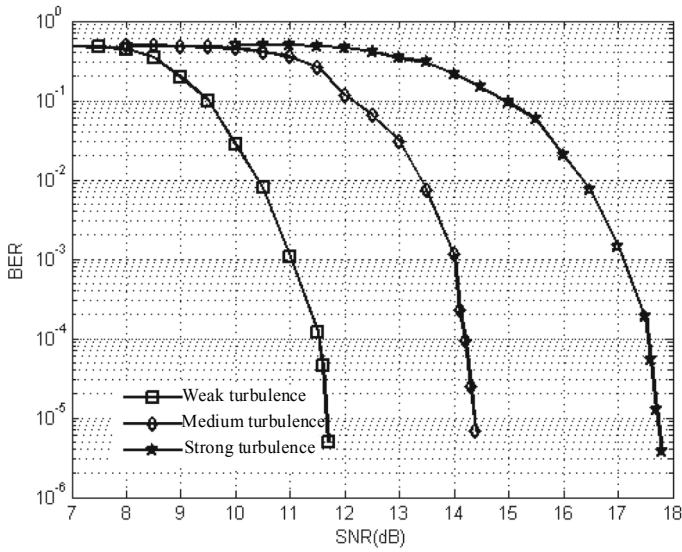


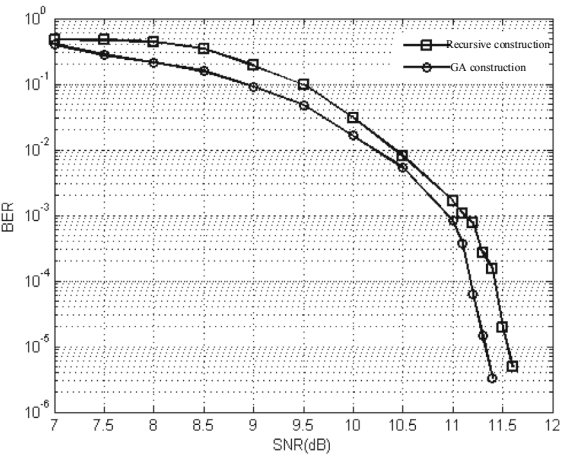
Fig. 5.4 BER performance curve of a recursively constructed polar code in an atmospheric-turbulence channel

additive Gaussian white noise on the channel polarization in an atmospheric channel. Therefore, this method is only empirical.

Figure 5.5 shows the decoding performance curve of the simulations of two different construction methods to realize polar codes under weak turbulence conditions. The code length is 1024 and the code rate is 0.5. It can be seen from the performance curve in Fig. 5.5 that at a 10^{-5} bit-error rate, the polar code constructed using the GA method in this study can obtain about a 0.2-dB coding-gain improvement compared with the recursive construction method. This is because the multiplicative noise in a turbulent channel and the influence of Gaussian noise on the channel polarization are considered, when using a Gaussian approximation to construct a polar code in a weak turbulent channel.

However, this construction method is based on the equivalence of the lognormal distribution of weak turbulence to a Gaussian distribution, which itself is an approximation. These approximations affect the code performance. If a polar code can be constructed for a weak turbulent channel, the performance will be better than that of existing ones.

Fig. 5.5 BER performance curves of two different construction methods



5.1.4 Polar-Code Construction in a Weak Atmospheric-Turbulence Channel

According to the channel Bhattacharyya parameter $Z(W_N^{(i)})$, calculated in the previous section, we select K (K is the dimension of the codeword space) that minimizes Eq. (5.12), and the set of index i is information-bit set \mathcal{A} . Here, the Bhattacharyya parameters calculated in Eq. (5.39) are sorted from small to large, and the index numbers corresponding to the first K ($K = N \times R$) values are selected to form the information-bit set \mathcal{A} . N and R are the code-length and code-rate parameters of the constructed polar code, respectively.

For the weak atmospheric-turbulence channel model described in Eq. (5.22), when the lognormal variance is $\sigma_0 = 0.2$, the polar codes have code lengths $N = 512, 1024$, and 2048 ; the bit-rate parameters R are $0.25, 0.5$, and 0.75 , as shown in Table 5.2. Table 5.2 also gives the reference values of the Bhattacharyya parameter threshold of polar codes under different code lengths and code rates.

According to the Bhattacharyya parameter value $Z(W_N^{(i)})$ calculated in Eq. (5.39), a set \mathcal{A} of index numbers i that is smaller than the corresponding reference threshold is selected, to determine the equivalent-coding matrix $G_N(\mathcal{A})$ of the polar code. We set the frozen bits $u_{\mathcal{A}^c}$ corresponding to the set \mathcal{A}^c as all-zero sequences and, according

Table. 5.2 Example of polar-code parameters

Code length N	512	1024	1024	1024	2048
Bit rate R	0.5	0.25	0.5	0.75	0.5
$Z(W_N^{(i)})$ threshold	0.88089	0.00010	0.61896	0.98415	0.57883

to Eq. (5.2), we can find the polar-coded output sequence x_1^N corresponding to the input information sequence u_A .

From Table 5.2, under different code-length and code-rate parameters, the Bhattacharyya parameter threshold of the virtual channel of a weak atmospheric-turbulence channel after merging and splitting can be seen. In the case of a short code length ($N = 512$) or high code rate ($R = 0.75$), the proportion of incomplete polarization selected in the polar-code construction process is already on the high side, and will inevitably deteriorate the performance of a polar code with a limited code length.

However, when the bit rate is low ($R = 0.25$) and the code length is long ($N = 2048$), the proportion of incomplete polarization selected in the construction process is low, and its error-correction performance should be good. The performance simulation results in Sect. 5.1.2 will verify the above phenomena and conclusions.

5.2 Performance of Polarization Code in Wireless Optical Channel

5.2.1 SC Decoding Algorithm

One standard to check whether coding performance is good or bad is to analyze the BER performance through the corresponding decoding algorithm. The serial-cancellation (SC) decoding algorithm is the first decoding algorithm used in polar codes, which were first proposed by E. Arikan [13]. The algorithm decodes bit by bit in sequence, and the decoder of channel $W_N^{(i)}$ uses the channel-received value and the decoding result of the previous bits to decode.

Introduction of the SC decoding algorithm

The channel $W_N^{(i)} : \mathcal{X} \rightarrow \mathcal{Y}^N \times \mathcal{X}^{i-1}$ obtained through channel merging and separation can be regarded as input u_i and output (y_1^N, u_1^{i-1}) , where y_1^N is the receiving sequence and u_1^{i-1} is the first $i-1$ information bits. The task of decoder i is to provide the estimated value \hat{u}_i of u_i , according to the received value y_1^N and the decoding results \hat{u}_1^{i-1} of the first $i-1$ decoders.

Assuming that the decoded sequence is \hat{u}_1^N , for $i, i = 1, 2, \dots, N$, if $i \in \mathcal{A}^C$, that is, i belongs to the frozen-bit set, because this part of the information is known to both the sending and receiving ends, the decoding process is simplified to only the $i \in \mathcal{A}^C$ part; hence, $\hat{u}_i = u_i$.

We define the log-likelihood ratio of the i^{th} bit u_i as

$$L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) \triangleq \ln \frac{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1} | 0)}{W_N^{(i)}(y_1^N, \hat{u}_1^{i-1} | 1)}. \quad (5.40)$$

Here, a hard judgment is adopted, and the judgment conditions are as follows:

$$\hat{u}_i = \begin{cases} 0 & L_N^{(i)}(y_1^N, \hat{u}_1^{i-1}) \geq 0 \\ 1 & \text{otherwise} \end{cases}. \quad (5.41)$$

The complexity of this algorithm is essentially determined by the complexity of calculating the log-likelihood ratio. In a concrete calculation, the following two recursions can be used for calculation:

$$\begin{aligned} L_N^{(2i-1)}(y_1^N, \hat{u}_1^{2i-2}) &= \frac{\exp\{L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,0}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}) + L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2})\} + 1}{\exp\{L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,0}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2})\} + \exp\{L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2})\}} \\ &= L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,0}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}) \boxed{+} L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}) \end{aligned} \quad (5.42)$$

$$L_N^{(2i)}(y_1^N, \hat{u}_1^{2i-1}) = L_{N/2}^{(i)}(y_{N/2+1}^N, \hat{u}_{1,e}^{2i-2}) + (-1)^{\hat{u}_{2i-1}} L_{N/2}^{(i)}(y_1^{N/2}, \hat{u}_{1,0}^{2i-2} \oplus \hat{u}_{1,e}^{2i-2}) \quad (5.43)$$

Among them, $\hat{u}_{1,0}^{2i-2}$ and $\hat{u}_{1,e}^{2i-2}$ represent odd and even-index sub-vectors in \hat{u}_1^{2i-2} , respectively. $\boxed{+}$ is called the box-plus operation [11] and is defined as

$$L_1 + L_2 \triangleq \log\left(\frac{1 + e^{L_1 + L_2}}{e^{L_1} + e^{L_2}}\right). \quad (5.44)$$

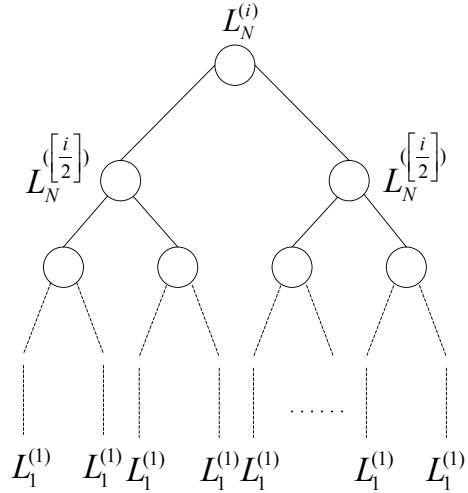
Therefore, the calculation of LLR with code length N is transformed into the calculation of two LLRs with code lengths of $N/2$; this recursion continues until $N = 1$, which can be directly calculated by defining $L_1^{(1)}(y_i) = \log(W(y_i|0)/W(y_i|1))$. In a weak atmospheric-turbulence channel, using OOK modulation,

$$L_1^{(1)}(y_i) = \log \frac{\frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{y_i^2}{2\sigma^2}\right)}{\frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{(y_i - I_i)^2}{2\sigma^2}\right)} = \frac{I_i^2 - 2y_i I_i}{2\sigma^2}. \quad (5.45)$$

Implementation of SC decoding algorithm

The recursive process of calculating the log-likelihood ratio $L_N^{(i)}(y_1^N, \hat{u}_1^{i-1})$ can be represented by a binary tree, as shown in Fig. 5.6. If the calculation of each node in the tree is the calculation amount of one unit, the calculation amount required to calculate $L_N^{(i)}(y_1^N, \hat{u}_1^{i-1})$ is the number of all nodes in the tree. For polar codes with parameters $(N, K, \mathcal{A}, u_{\mathcal{A}})$, the amount of computation required to calculate $L_N^{(i)}(y_1^N, \hat{u}_1^{i-1})$ is $2N - 1$. Because the log-likelihood ratio of frozen bits does not

Fig. 5.6 Recursive SC-decoding process



need to be calculated, the total amount of computation is $K(2N - 1) = RN(2N - 1)$ and R is the code rate; thus, the asymptotic complexity of the decoding is $O(N^2)$.

Observing Eqs. (5.42) and (5.43), in the recursive-calculation formula, the two parts to be calculated include the same terms, so there is no need to repeat the calculation; this can greatly reduce the decoding complexity. Based on the above observation, the decoding trees of all bits can be combined into one graph, so the intermediate-calculation result only needs to be calculated once.

The combined-grid diagram is shown in Fig. 5.7, in which the intermediate results $L_N^{(i)}(y_1^N, u_1^{i-1})$ of the recursive calculation are represented by nodes marked as (y_1^N, u_1^{i-1}) . In this case, the total number of nodes to be calculated in the figure is $N(\log N)$; thus, the asymptotic complexity of the SC decoding is only $O(N \log N)$.

It is noted that the calculation of each node requires the calculation result of its successor node, so decoding u_i is essentially a process of traversing a tree in post-order. In the traversal process, each node needs to pass the translated information to its successor node and wait for the successor node to complete the calculation; finally, it calculates according to Eqs. (5.42) and (5.43), and passes the result back to its parent node.

In summary, the specific process of the SC decoding algorithm is as follows:

1. Initialize the last column of nodes in the grid diagram with the received value, according to Eq. (5.45);
2. For $i = 1, \dots, N$, execute steps 3 and 4;
3. Take the i^{th} node in the first column of the grid graph as the root node for post-order traversal;
4. If $i \in \mathcal{A}$, then, \hat{u}_i is obtained according to the judgment conditions of Eq. (5.41); otherwise, \hat{u}_i is judged according to the frozen-bit set;
5. Output the final decoding result.

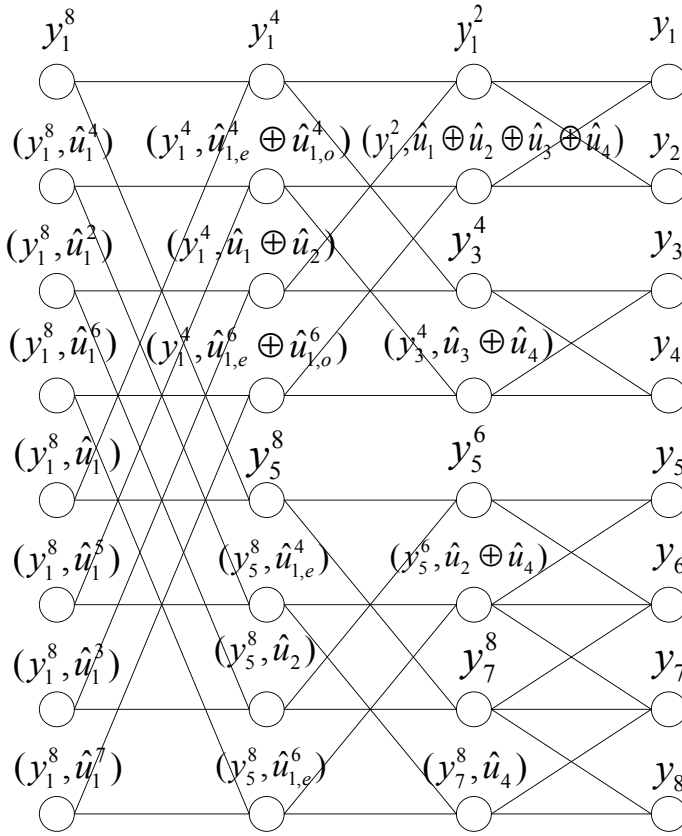


Fig. 5.7 SC decoding grid

In this decoding process, owing to the use of the recursive operation, the box-plus operation is frequently used in Eq. (5.42), which includes exponential and logarithmic floating-point-numbers operations, and the calculation complexity is high. Therefore, the execution speed of this operation will have a significant impact on the decoding efficiency, and it is not easy to realize in practice.

Another expression of the box-plus operation is given in reference [15]:

$$L_1 + L_2 = \max(0, L_1 + L_2) + s(L_1, L_2). \quad (5.46)$$

Here, the correction term is

$$s(x, y) = \log(1 + e^{-|x+y|}) - \log(1 + e^{-|x-y|}). \quad (5.47)$$

In Eq. (5.46), except for $s(L_1, L_2)$, there are no exponential or logarithmic operations, which can be calculated conveniently. Note that these two items in $s(x, y)$ have similar expressions, which can be recorded as $g(x) = \log(1 + e^{-|x|})$. Then,

$$s(x, y) = g(x + y) - g(x - y) \quad (5.48)$$

Therefore, the calculation of $s(x, y)$ can be transformed into the calculation of $g(x)$. Because the function value of $g(x)$ has a small variation range ($0 < g(x) \leq \log 2$), and the function value decays rapidly with the increase of independent variables, the table-lookup method can be used to calculate it.

It should be noted that this table-lookup method will introduce errors into the decoding, and these errors will accumulate in the recursive process. For a larger code length, the error accumulation will become more obvious with more layers in the decoding tree. Therefore, with the increase of code length, the accuracy required for the table lookup will be improved accordingly. However, because the table construction is discrete and linear, improving the table-lookup accuracy only increases the spatial complexity, not the time complexity.

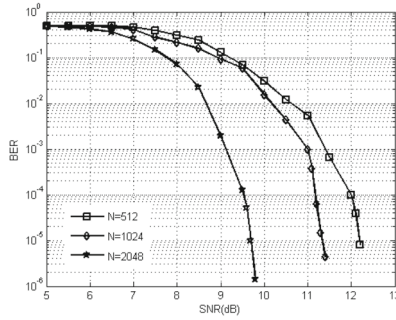
Although the SC-decoding algorithm has a low algorithmic complexity, the decoding result of each bit can only be used by the subsequent decoder. If an incorrect decision is made, it cannot be found and corrected in the subsequent decoding. Moreover, the input to the subsequent decoder is the hard decision of the decoding result, which inevitably loses some information; that is, the decoding result of each bit is not fully utilized by the subsequent decoder. Therefore, some scholars have improved the SC decoding algorithm to improve its performance of SC decoding [16–19].

Monte-Carlo simulation of the SC decoding-algorithm performance

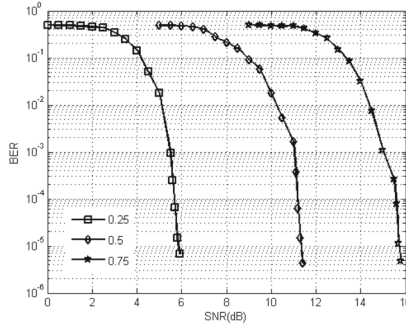
The decoding and error-correction performance of the SC algorithm in different intensity atmospheric-turbulence channels is analyzed using Monte-Carlo simulation. The code lengths of the polar codes are 512, 1024, and 2048, and the code rates are 0.25, 0.5, and 0.75. The modulation method is OOK.

Figure 5.8 shows the decoding simulation results of the polar-code SC algorithm in a weak turbulent channel constructed using Gaussian approximation. It can be seen from the figure that, for a large signal-to-noise ratio, the bit-error rate of the polar code in a weak turbulent channel can reach 10^{-6} or even lower. Figure 5.8(a) shows the comparison of the bit-error rate performance with the same code rate, $R = 0.5$, and different code lengths, $N = 512, 1024, 2048$.

It can be seen from the figure that when the code rate is the same, the performance of the polar code improves with the increase of the code length. The polar code with a 1024 code length has an improvement of about 0.8 dB, compared with the polar code with a 512 code length, while the polar code with a 2048 code length has about a 1.5-dB improvement over the polar code with a 1024 code length. It can be seen that increasing the code length can improve the performance of polar codes. Although this simultaneously increases the overhead and delay of the encoder and decoder, such a sacrifice is acceptable, owing to the low asymptotic complexity of coding and decoding (only $O(N \log N)$).



(a) Code rate is $R = 0.5$, code lengths are $N=512, 1024, 2048$



(b) Code length is $N = 1024$, code rates are $R=0.25, 0.5, 0.75$.

Fig. 5.8 Polar-code performance curve in a weak turbulence channel constructed using GA

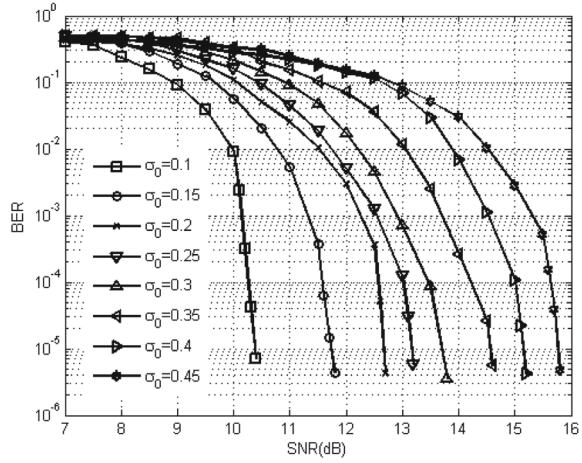
Figure 5.8b shows the impact of different code rates on polar codes when the code length is the same. From the figure, we can intuitively see that the code rate has a great impact on polar codes. This is because, with the increase of the code rate, more channels that are not fully polarized are used for information transmission. The influence of noise on these channels becomes more and more obvious with the increase of the code rate, which increases the bit-error rate and reduces the decoding performance.

- (a) Code rate is $R = 0.5$, code lengths are $N = 512, 1024, 2048$.
- (b) Code length is $N = 1024$, code rates are $R = 0.25, 0.5, 0.75$.

In the case of a weak turbulence channel constructed by a Gaussian approximation, the influence of weak turbulence noise on the communication system when the code length is $N = 1024$ and the code rate is $R = 0.5$ is shown in Fig. 5.9. The figure shows the simulation curve of the bit-error rate decoded by the SC algorithm under different turbulence intensities in the case of weak turbulence.

From the final simulation diagram, it can be seen that the performance of the system is reduced, owing to the introduction of turbulence noise, and this performance is more and more obvious with the increase of turbulence intensity. This is

Fig. 5.9 Effect of different turbulence intensities on the bit-error rate



because turbulent noise affects the channel polarization. When the code rate and code length are the same, the larger the turbulent noise, the fewer channels are completely polarized, which leads to an increase in the bit-error rate. Therefore, in the case of a large turbulence noise, the system performance can be improved by reducing the bit rate; thus, ensuring the reliability of the transmission.

5.2.2 Performance Analysis of Polar Codes Under PPM Technology

For optical wireless communication, two methods are mainly considered: on–off keying (OOK) and pulse-position (PPM) modulation. The influence of PPM modulation technology on the performance of polar codes in an atmospheric-turbulence channel is mainly discussed.

(1) Introduction of PPM Modulation to an Atmospheric-Turbulence Channel

In the case of weak atmospheric turbulence, according to the channel model analysis, the mathematical model of an atmospheric-turbulent optical-communication channel is

$$y_k = I_k z_k + n_k I_k > 0, 1 \leq k \leq M. \quad (5.49)$$

Here, $z_k \in \{0, 1\}$ is the signal from the k time slot of the j th PPM frame, y_k is the soft-output information received after the coded signal is transmitted through the channel, n_k is the additive Gaussian white noise with a mean value of 0 and variance of $N_0/2$, I_k is the light intensity satisfying a lognormal distribution, n_k and I_k are

independent of each other, and M is the number of time slots in an information frame.

When an IM/DD atmospheric-turbulence communication system adopts PPM modulation, it is assumed that the light intensity I is known, as well as a fixed value within the information-bit time interval of each time slot transmitting the j th PPM frame. Then, in the time interval of each time slot of the j th PPM frame, the lognormal fading channel is equivalent to a random binary input $\{0, I\}$, continuous output memoryless Gaussian channel.

At this time, the mathematical model of a weak atmospheric-turbulence channel is equivalent to

$$y_d = I_d + n_d, y_k = n_k (k \neq d, 1 \leq k \leq M, 1 \leq d \leq M). \quad (5.50)$$

Here, y_d represents the pulse position in the j th PPM frame, and k represents the non-pulse position. Therefore, the soft-output signal y_k received from the k th slot of the j th PPM frame only comes from additive Gaussian noise; that is, the conditional distribution $P(y_k | z_k = 0)$ obeys the Gaussian distribution with a mean value of 0 and variance of $N_0/2$. Therefore, the conditional probability-density function model is

$$P(y | z = 0) = \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{y^2}{N_0}\right). \quad (5.51)$$

The soft output signal y_d received from the d th slot of the j th PPM frame comes from the instantaneous light intensity I and additive Gaussian noise; that is, the conditional distribution $P(y_d | z_d = 1)$ obeys the Gaussian distribution with a mean value of I_d and variance of $N_0/2$. Therefore, the conditional probability-density function model is

$$P(y | z = 1) = \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{(y - I)^2}{N_0}\right). \quad (5.52)$$

Each time slot in each PPM frame can be regarded as OOK modulation. In addition, in the PPM frame of each M slot, only one slot has light intensity, while the remaining $M - 1$ slots have no light intensity. Assuming that the received soft outputs are independent of each other, d represents the position with light intensity. The soft-output probability-density function of the time slot with light intensity is represented by $p_s(\cdot)$ and the soft-output probability-density function of a time slot without light intensity is represented by $p_n(\cdot)$; then,

$$p_s(y_i) = \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{(y_i - I_i)^2}{N_0}\right). \quad (5.53)$$

$$p_n(y_i) = \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{y_i^2}{N_0}\right). \quad (5.54)$$

Therefore, the log-likelihood ratio of the i^{th} slot of the j^{th} PPM frame is

$$L_i^j = \frac{p_n(y_i)}{p_s(y_i)} = \exp\left(\frac{I_i^2 - 2y_i I_i}{N_0}\right). \quad (5.55)$$

Because there are M time slots in a PPM frame, it corresponds to $A = M / (\log_2 M)$ codes. Therefore, the posterior probability of each symbol is expressed as [12]

$$P(c) = \frac{\sum_{\substack{\mathbf{X}=(x_1, x_2, \dots, x_A) \\ x_l=c}} L_d^j}{\sum_{i=1}^M L_i^j}. \quad (5.56)$$

Here, $c = 0, 1$, the vector $\mathbf{X} = (x_1, x_2, \dots, x_A)$ represents the symbol corresponding to the j^{th} PPM frame, and $1 \leq l \leq A$ is the currently calculated symbol. Therefore, the log-likelihood ratio at this time can be calculated by the following formula:

$$L(l) = \ln\left(\frac{P(0)}{P(1)}\right) = \ln\left(\sum_{\substack{\mathbf{X}=(x_1, x_2, \dots, x_A) \\ x_l=0}} L_d^j\right) - n\left(\sum_{\substack{\mathbf{X}=(x_1, x_2, \dots, x_A) \\ x_l=1}} L_d^j\right). \quad (5.57)$$

(2) PPM performance simulation of polar coding

Figure 5.10 shows a simulation diagram of the bit-error rate performance of polar codes constructed by Gaussian approximation using OOK modulation and 2-PPM modulation under weak atmospheric turbulence, with bit rates of 0.5. As can be seen from the curves in Fig. 5.10, polar codes in the atmospheric-turbulence channel can be significantly improved by using 2-PPM modulation, compared with OOK modulation. Under the same code-length and code-rate parameters, the polar-coding modulation scheme with 2-PPM modulation can obtain a performance gain of about 3.5 dB to 3.8 dB, compared with the polar-coding modulation scheme with OOK modulation.

As the number of time slots m of the M -PPM modulation increases, the performance simulation results of the PPM-coding modulation scheme of the constructed polar code under an atmospheric-turbulence channel are shown in Fig. 5.11. It can be seen that the demodulation, decoding, and error-correction performance decreases with the increase of the number of time slots M ; at a 10^{-6} bit-error rate, the performance loss of 4-PPM is about 1.3 dB, compared with 2-PPM, and the performance of 16-PPM modulation is worse than that of OOK modulation.

This is because each M -PPM frame has M time slots. With the increase of the number of time slots, the probability of a demodulation error in each frame increases accordingly, and the increase of the number of time slots M will bring more turbulence and Gaussian noise, which will reduce its demodulation and decoding performance.

Fig. 5.10 BER performance of PPM-modulated polar codes with different code lengths

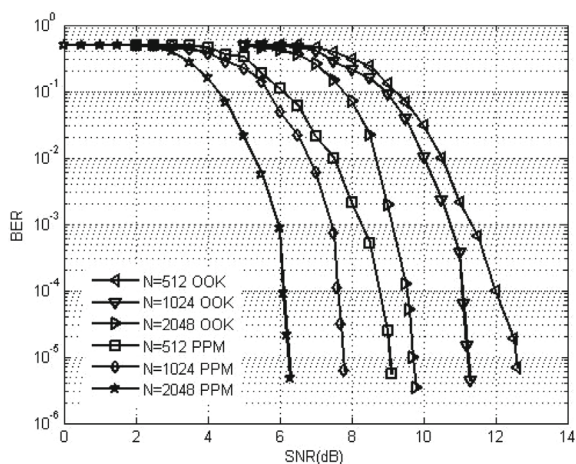
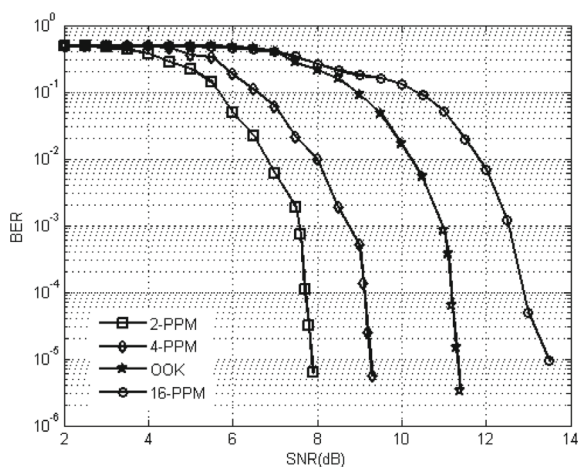


Fig. 5.11 Effect of time-slot number on the bit-error rate in PPM modulation



However, with the increase of the number of time slots, the average laser power required by the transmitting end decreases, and the higher-order PPM modulation has higher power utilization, which can improve the effective communication distance of optical wireless communication. In practical-application scenarios, it is necessary to make a reasonable compromise.

5.2.3 Performance of Polar Codes Under Subcarrier Modulation

Subcarrier-intensity modulation is a modulation technology that can effectively overcome atmospheric turbulence. It has been confirmed in references [17] and [18] that the performance of subcarrier modulation is better than OOK in an atmospheric-turbulence environment. In this section, the polar code is introduced as the channel-coding method for optical wireless communication, and the bit-error performance of the system using polar codes under subcarrier modulation is discussed for a weak atmospheric-turbulence channel.

The bit-error performance of a polar-code system based on subcarrier binary phase-shift keying (BPSK) and quadrature phase-shift keying (QPSK) modulation in a weak turbulent channel is analyzed by simulation. Finally, the bit-error rate (BER) of the polar code measured by subcarrier modulation is analyzed under three different weather conditions.

(1) Subcarrier Modulation in an Atmospheric-Turbulence Channel

The block diagram of a subcarrier IM/DD optical wireless communication system based on polar codes is shown in Fig. 5.12 [17].

The information sequence generated by the source is encoded by polar code, then modulated by subcarrier, and then the light source is intensity modulated and enters the atmospheric channel through the transmitting antenna. The receiving end performs a photoelectric conversion through an avalanche photo-diode (APD) photodetector, then restores the information sequence through corresponding subcarrier demodulation and polar-code decoding, and finally calculates the bit-error rate.

When the polar code is coded and modulated by M -phase-shift keying, the input binary sequence is converted through a serial-parallel conversion, and the symbol of $\log_2 M$ bits is mapped into one symbol; then, the symbol can be represented on the constellation through $s^m = s_x^m + js_y^m$ ($m = 1, \dots, \log_2 M$). The mathematical expression is

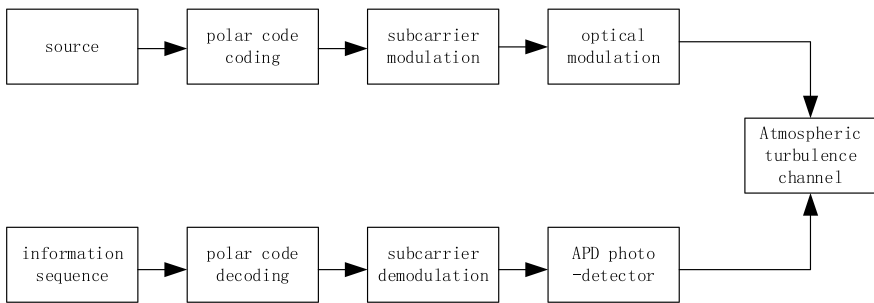


Fig. 5.12 Subcarrier optical wireless communication system based on polar codes

$$s(t) = A_0 \cos(\omega_c t + \varphi_n) = I(t) \cos \omega_c t - Q(t) \sin \omega_c t, \quad (5.58)$$

where the in-phase component is $I(t) = A_0 \cos(\varphi_n)$ the orthogonal component is $Q(t) = A_0 \sin(\varphi_n)$, and $\varphi_n = \arg^{-1} \frac{I(t)}{Q(t)}$.

Assuming that the transmission symbol is $S_i = x_T^{S_i} + jy_T^{S_i}$, $x_T^{S_i}$ and $y_T^{S_i}$ represent the in-phase component I and quadrature component Q of the transmission symbol, respectively. The receiving symbol is $\hat{S}_i = x_R^{S_i} + jy_R^{S_i}$, and $x_R^{S_i}$ and $y_R^{S_i}$ represent the in-phase component I and quadrature component Q of the transmitting symbol, respectively. For I and Q orthogonal signals,

$$P(x_R^{S_i} | x_T^{S_i}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_R^{S_i} - x_T^{S_i})^2}{2\sigma^2}\right) \quad (5.59)$$

$$P(y_R^{S_i} | y_T^{S_i}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_R^{S_i} - y_T^{S_i})^2}{2\sigma^2}\right). \quad (5.60)$$

σ^2 represents the variance of Gaussian noise.

Assuming that symbol S_i is transmitted with equal probability and the received symbol is known, the posterior probability of the transmitted symbol is changed to

$$\begin{aligned} P\{S_i | \hat{S}\} &= P(x_T^{S_0} | x_R) P(y_T^{S_0} | y_R) = K_x K_y P(x_R | x_T^{S_0}) P(y_R | y_T^{S_0}) \\ &= K \exp\left(-\frac{\left((x_T^{S_0})^2 + (y_T^{S_0})^2\right) + ((x_R)^2 + (y_R)^2) - 2(x_T^{S_0} x_R + y_T^{S_0} y_R)}{2\sigma^2}\right), \\ &= K \exp\left(\frac{2(x_T^{S_0} x_R + y_T^{S_0} y_R) - ((x_R)^2 + (y_R)^2) - 1}{2\sigma^2}\right) \end{aligned} \quad (5.61)$$

where K is a common factor.

For a QPSK modulation system, one symbol represents two-bit symbols, and "00," "01," "10," and "11" represent four phases. If the currently received symbol is \hat{S} , the probabilities that the first bit symbol is "0" or "1" are

$$P(0 \times | \hat{S}) = P(S_0 | \hat{S}) + P(S_1 | \hat{S}) \quad (5.62)$$

$$P(1 \times | \hat{S}) = P(S_2 | \hat{S}) + P(S_3 | \hat{S}). \quad (5.63)$$

In addition, the probability of other bit symbols can also be calculated. From Eqs. (5.62) and (5.63), it can be deduced that the log-likelihood ratio of the first bit symbol is

$$L(q) = \ln \frac{P(0)}{P(1)} = \ln \frac{P(0|\hat{S})}{P(1|\hat{S})} = \ln \left(\frac{\sum_{i \in N_0} P\{S_i|\hat{S}\}}{\sum_{i \in N_1} P\{S_i|\hat{S}\}} \right), \quad (5.64)$$

where $N_0 \in \{0, 1\}$ and $N_1 = \{2, 3\}$. Therefore, the log-likelihood ratio of all symbols can be calculated according to the received symbols, to decode the polar-code SC algorithm. Similarly, for an MPSK modulation system, the log-likelihood ratio of symbols can also be calculated by this method.

(2) Simulation and Experimental Analysis Under Subcarrier Modulation

In the simulation, the SC algorithm is the polar-code decoding algorithm, the code length is 512, and the code rate is 0.5. Figure 5.13 shows the bit-error rates of an uncoded optical wireless communication system with subcarrier BPSK and QPSK modulation under different turbulence-intensity channels ($\sigma_0 = 0.1, \sigma_0 = 0.2$, and $\sigma_0 = 0.3$) and different signal-to-noise ratios (SNRs).

Figures 5.14 and 5.15 show the bit-error rate performance of an optical wireless communication with subcarrier BPSK modulation based on polar codes and a QPSK-modulation system under different turbulence-intensity channels, respectively. Compared with Fig. 5.12, it can be seen that the performance of the subcarrier system encoded using polar codes is better than that of the uncoded subcarrier system, and the BPSK system encoded using polar codes is better than that of QPSK, and a coding gain of 4.8 dB can be obtained.

Figure 5.16 is an analysis of some measured data from an optical-wireless subcarrier modulation system based on polar codes. Figure 5.16(a) shows the demodulation

Fig. 5.13 Bit-error rates of an uncoded subcarrier system

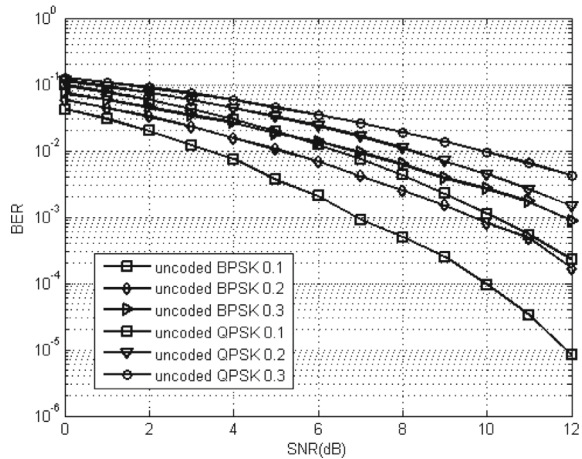


Fig. 5.14 Bit-error rates of a BPSK subcarrier system based on polar codes

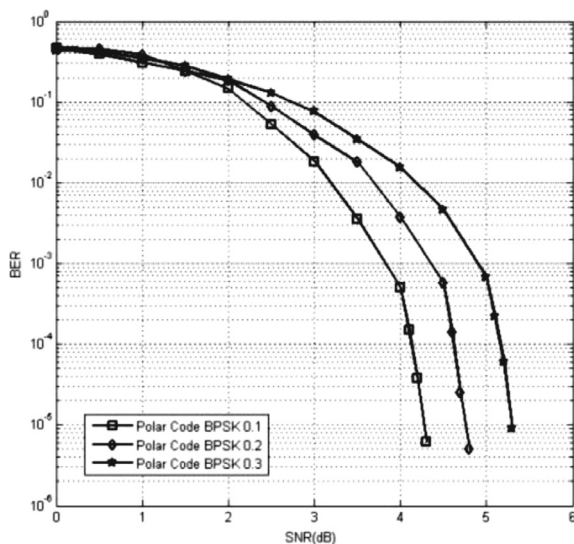
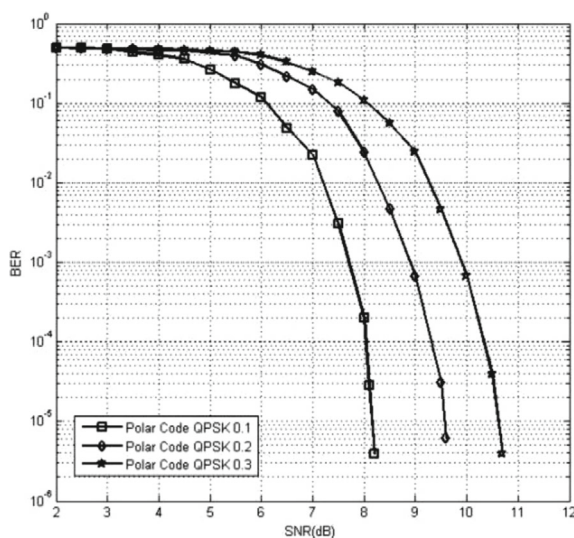


Fig. 5.15 Bit-error rates of a QPSK subcarrier system based on polar codes



sequence obtained after demodulating the received signal, in which the underlined parts represent bit errors generated after demodulation, and the information sequence is decoded as the input of the SC polar-code decoding algorithm.

Figure 5.16(b) is the decoded output sequence, in which the underlined part is the information sequence recovered from the information-bit set. Compared with the original input sequence, there are no bit errors. Therefore, the error caused by the demodulation of the signal received by the system was corrected.

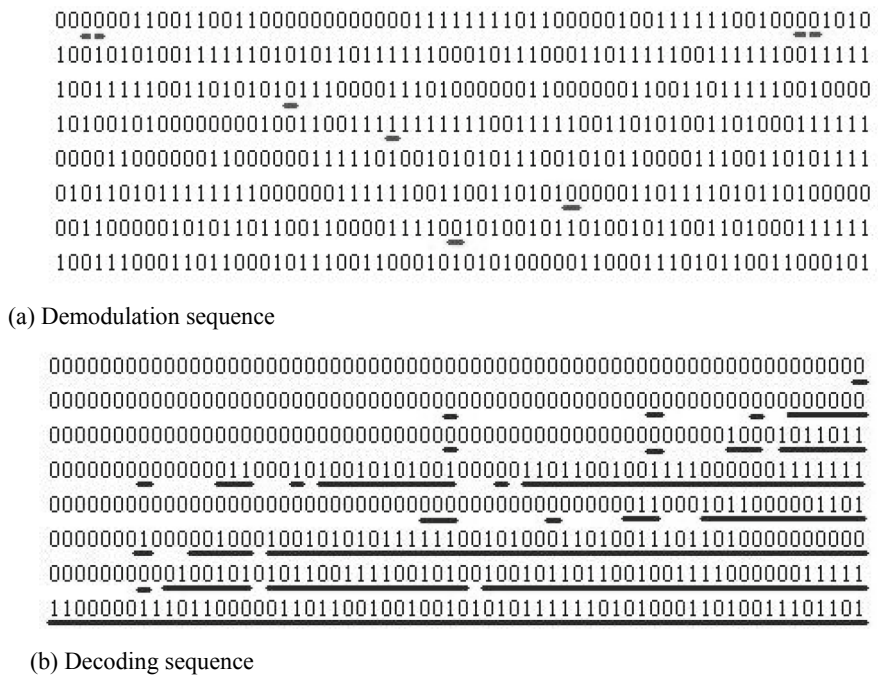


Fig. 5.16 Analysis of measured data of an optical-wireless subcarrier-modulation system based on polar codes

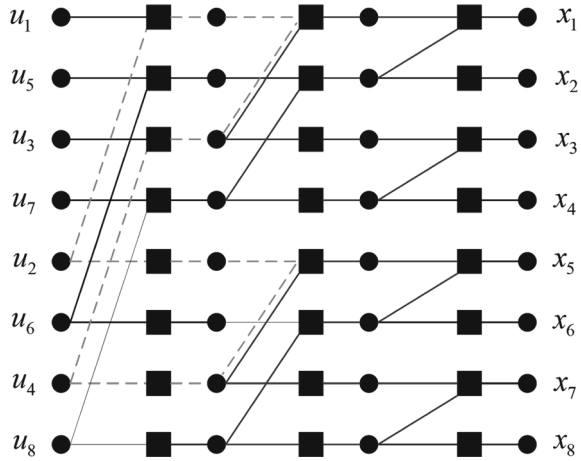
- (a) Demodulation sequence.
- (b) Decoding sequence.

Table 5.3 is an analysis of the measured data error rates of an uncoded optical-wireless subcarrier-modulation system and an optical-wireless subcarrier-modulation system based on polar codes under three weather conditions (sunny, cloudy, and rainy). Through a comparative analysis of the demodulation bit-error rate and decoding bit-error rate in the measured data, it is shown that in the turbulent channel, the bit-error rate characteristics can be improved by an order of magnitude by using polar-code coding technology, to effectively improve the reliability of the optical-wireless system.

Table 5.3 Analysis of measured data from an optical-wireless subcarrier-modulation system

	Sunny	Cloudy	Rainy
Transmitted symbols (number)	10,240	12,288	13,824
Demodulation errors (number)	122	12	147
Decoding errors (number)	12	0	18
Demodulation bit-error rate	0.0119	0.000977	0.0106
Decoding error rate	0.00234	0	0.00261

Fig. 5.17 Rings in polar-code factor graphs



5.2.4 Comparison Between Polar Codes and LDPC Codes

Polar codes and low-density parity-check (LDPC) codes have many similarities. First, they are linear error-correction codes. Second, LDPC codes have a low-density check matrix, while polar codes have a low-density generation matrix. In reference [20], a polar-code check-matrix construction method is proposed. The check matrix constructed by this method also has a low density, so it can be decoded using the backpropagation algorithm.

In terms of code-performance, polar codes have been mathematically proven to be able to reach the Shannon limit with low coding and decoding complexity; however, there is no proof for LDPC codes at present. From the simulation effect of the design, to achieve the same rate and the same error probability as LDPC codes, polar codes need longer code lengths; however, they can be realized with lower complexity.

On the other hand, polarimetric codes have better error leveling. Some scholars have simulated polarimetric codes in BEC channels and found that there is no error leveling when the bit-error rate is 10^{-11} . The literature points out that the reason why polar codes have such excellent error leveling is that they have a good stopping distance. In addition, the minimum ring length (girth) in the factor graph of a polar code is 12, which is also considered to be very ideal in LDPC codes [18] (Fig. 5.17).

References

1. Arkan E (2008) A performance comparison of Polar Codes and Reed-Muller codes. *Commun Lett, IEEE* 12(6):447–449
2. Dongfeng Y, Haixia Z (2006) Advanced channel coding technology in broadband mobile communication. Beijing University of Posts and Telecommunications Press, Beijing

3. Ke X, Zhiyun Y (2009) Coding theory in atmospheric laser communication system. Beijing, Science Press, pp 8–10
4. Ke X, Xiaoli X (2004) Introduction to wireless laser communication. Beijing, Beijing University of Posts and Telecommunications Press, p 10
5. Abbe E, Barron A (2011) Polar coding schemes for the AWGN channel. In: 2011 IEEE International Symposium on Information Theory Proceedings (ISIT). IEEE, pp 194–198
6. Bravo-Santos A. Polar codes for Gaussian degraded relay channels. 20s.
7. Sandalidis HG, Tsiftsis TA (2008) Outage probability and ergodic capacity of free-space optical links over strong turbulence. *Electron Lett* 44(1):46–47
8. Richardson TJ, Shokrollahi MA, Urbanke RL (2001) Design of capacity-approaching irregular low-density parity-check codes. *IEEE Trans Inf Theory* 47(2):619–637
9. Chung SY, Richardson TJ, Urbanke RL (2001) Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation. *IEEE Trans Inf Theory* 47(2):657–670
10. El Gamal H, Hammons AR Jr (2001) Analyzing the turbo decoder using the Gaussian approximation. *IEEE Trans Inf Theory* 47(2):671–686
11. Li J, Uysal M (2003) Optical wireless communications: system model, capacity and coding. In: Vehicular Technology Conference, VTC 2003-Fall. 2003 IEEE 58th. IEEE, 2003, no 1, pp 168–172
12. Li H, Yuan J (2013) A practical construction method for polar codes in AWGN channels. In: IEEE TENCON Spring conference, Sydney, pp 223–226
13. Arikan E (2009) Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. *IEEE Trans Inf Theory* 55(7):3051–3073
14. Zhao S, Shi P, Wang B (2011) Designs of Bhattacharyya parameter in the construction of Polar codes. In: 2011 7th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM). IEEE, pp 1–4
15. Ryan W, Lin S (2009) Channel codes: classical and modern. Cambridge University Press
16. Alamdar-Yazdi A, Kschischang FR (2011) A simplified successive-cancellation decoder for Polar Codes. *Commun Lett, IEEE* 15(12):1378–1380
17. Huang W, Takayanagi J, Sakanaka T et al (1993) Atmospheric optical communication system using subcarrier PSK modulation. *IEICE Trans Commun* 76(9):1169–1177
18. Lu Q, Liu Q, Mitchell GS (2004) Performance analysis for optical wireless communication systems using subcarrier PSK intensity modulation through turbulent atmospheric channel. In: Global Telecommunications Conference, 2004. GLOBECOM'04, no 3. IEEE, pp 1872–1875
19. Chen D, Ke X (2010) Performance analysis of subcarrier error in optical wireless communication based on Turbo codes. *Acta Optica Sinica* (10):2859–2863
20. Korada SB, Sasoglu E, Urbanke R (2010) Polar codes: characterization of exponent, bounds, and constructions. *IEEE Trans Inf Theory* 56(12):6253–6264

Chapter 6

Channel Measurements and Error-Control Experiments



It is generally believed that it is possible for a wireless laser-communication system to effectively realize all-weather communication within a certain sight distance. However, owing to the great differences in meteorological conditions, due to different geographical locations, landforms, environmental pollution, and other factors, laser atmospheric-communication systems are widely used in various regions and conditions. Improving the system performance under various climatic conditions is very important. Therefore, it is necessary to study the impact of atmospheric channels on system performance under various typical meteorological conditions [1–4].

6.1 Basic Laser-Transmission Theory in the Atmosphere

The atmosphere consists of a mixture of various gas molecules and suspended particles. Owing to temperature differences, wind, and other factors, molecules and particles in the atmosphere are in constant motion, and their composition, humidity, and density are constantly changing; thus, the atmosphere is in continuous turbulent motion.

The nature of the atmosphere has a significant impact on laser-beam transmissions. Atmospheric turbulence severely disturbs the beam because the atmospheric molecules and suspended particles absorb and scatter it. Atmospheric absorption mainly leads to the loss of beam energy and a weakening of the received signal, which is often called atmospheric attenuation in engineering. Scattering causes beam-intensity flicker, beam drift, expansion, and jitter, which are usually called the atmospheric-turbulence effect [5–8]. To ensure the normal and reliable operation of a laser-communication link, the influence of atmospheric attenuation and turbulence on laser transmissions should be considered in terms of atmospheric-channel transmission factors.

6.1.1 Energy Attenuation of a Laser Atmospheric Transmission [9–16]

When a laser propagates through the atmosphere, its energy gradually decreases. Atmospheric attenuation is primarily caused by the absorption and scattering of gas molecules, aerosols, and suspended particles in the atmosphere.

(1) Atmospheric-absorption effect

The atmosphere absorbs light. In the ultraviolet, visible, and infrared regions, the main absorption molecules are H_2O , CO_2 , O_3 , O_2 , and small amounts of CO , CH_4 , and N_2O . The absorption of a laser is determined by the characteristics of the atmospheric molecules' molecular-absorption spectra. A large number of absorption-spectral lines of gas molecules form a spectral-band group, which continuously absorbs light radiation. This absorption is weak in only a few wavelength regions, forming the so-called "atmospheric window."

Figure 6.1 shows the absorption of solar radiation by Earth's surface atmosphere [14]. The most important atmospheric windows are the visible-light band, 3–5- μm band, and 8–13- μm band. The absorption characteristics of atmospheric molecules are complex, and the attenuation of the received energy is closely related to the wavelength of the laser beam, which is at the 1.06- μm wavelength and the visible band.

The absorption of atmospheric molecules can typically be ignored; however, at the 10.6- μm wavelength, the atmospheric molecular absorption is the most severe. This absorption is also related to altitude because the closer to the ground (thousands of feet), the greater the water-vapor concentration and the more energy is absorbed by water vapor.

(2) Atmospheric-scattering effect

The diameter distribution of atmospheric particles is very wide, ranging from 10^{-4} microns to tens of microns. Particles with diameters ranging from 0.1 to 10 μm have

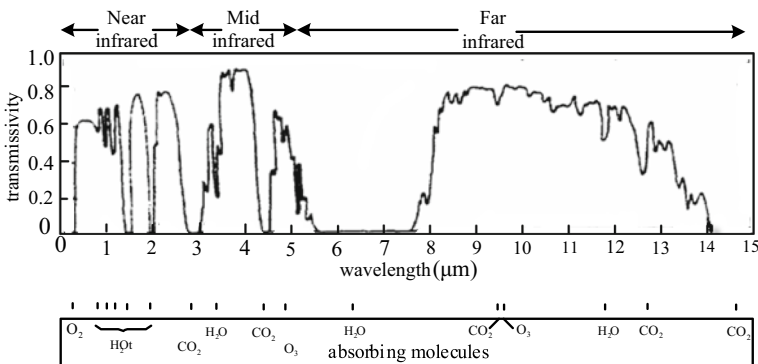


Fig. 6.1 Atmospheric transmittance [14]

the greatest impact on light transmission. We can use Rayleigh and Mie scattering from single-particle scattering theory to analyze the atmospheric scattering. This is because the spacing of gas molecules is more than ten times the molecular diameter, and the spacing of aerosol particles or suspended particles in the atmosphere is also much larger than a particle diameter; thus, it meets the conditions of single-particle scattering. When the particle spacing is more than three times the particle diameter, the particles hardly affect each other.

Therefore, although many types of particles exist in the atmosphere, the single-particle scattering theory is always applicable to light propagation. Moreover, the approximate mathematical treatment of single-particle scattering is simple and suitable for engineering applications. The attenuation coefficient of the laser energy generated by scattering is β :

$$\beta = \beta_m + \beta_p, \quad (6.1)$$

where β_m is the molecular-scattering coefficient, $\beta_m = \sigma_m \cdot n$; β_p is the particle-scattering coefficient, σ_m is the molecular-scattering cross section, which is inversely proportional to the wavelength, λ , and n is the molecular density of air.

a. Rayleigh scattering

Rayleigh scattering occurs when the light wavelength is larger than the size of the scattering particles. Rayleigh scattering mainly occurs in the ultraviolet-wave band or at high altitudes with few suspended particles. This scattering is directly proportional to the molecular density and inversely proportional to the fourth power of the scattered-light wavelength. Its scattering coefficient is [14]

$$\alpha_m(\lambda) = \frac{8\pi^3}{3} \cdot \frac{(n^2 - 1)^2}{N\lambda^4}, \quad (6.2)$$

where N is the number of particles per unit volume, λ is the wavelength, and n is the refractive index of the medium. The Rayleigh scattering coefficient in dry and fresh air is

$$\alpha_m(\lambda) = 1.09 \times 10^{-3} \lambda^{-4.05} \text{ km}^{-1}. \quad (6.3)$$

It can be seen that the scattering coefficient is inversely proportional to the fourth power of the wavelength. The smaller the wavelength, the stronger is the scattering. A clear sky is blue, rather than the color of sunlight, because the atmosphere has a large scattering coefficient for blue light waves.

b. Mie scattering

Mie scattering occurs when the particle size and wavelength are close to the same size. Mie scattering is applicable to spherical particles, such as light rain, fog droplets, and haze. Its size is roughly inversely proportional to the first power of the scattered

light and is also related to the particle-size distribution, height, distance, and other natural factors. Its scattering coefficient is

$$\alpha_m = \pi N(r) Q_s(X_r, m) r^2, \quad (6.4)$$

where $N(r)$ is the number of particles per unit volume (cm^3), R is the particle radius (cm), Q_s is the scattering efficiency factor, which is defined as the ratio of the scattered-particle energy to the energy of the geometric section of the incident particle. It is a function of the relative scale, $X_r = 2\pi r/\lambda$, of the particles and the complex refractive index $m = n - iK_a$, n and K_a are the real and imaginary parts of the complex refractive index, respectively. A detailed Q_s expression can be found in Chap. 3 of Ref. [15].

When a continuous distribution of particles exists between radii r_1 and r_2 , the Mie scattering coefficient is obtained from the following integral [7]:

$$\alpha_m = \pi \int_{r_1}^{r_2} n(r) Q_s(X_r, m) r^2 dr, \quad (6.5)$$

where $n(r)$ is the number of particles with a radius distributed in $[r, r + dr]$ per unit volume.

In general, for particles with radius $r \leq 0.03 \mu\text{m}$, if the wavelength is near $1 \mu\text{m}$, the error of Rayleigh scattering is $\leq 1\%$. When the particle radius $r > 0.03 \mu\text{m}$, the Mie scattering theory needs to be used. In the visible-light to infrared-light wavelength range (approximately $10 \mu\text{m}$), the scattering coefficient of raindrops has little relationship to the wavelength, and the attenuation coefficient per kilometer is [16]

$$\alpha_J = 0.21 J^{0.74}, \quad (6.6)$$

where J is the rainfall rate (mm/h).

(3) Combined effects of scattering and absorption

a. Atmospheric transmittance

A laser is transmitted in the atmosphere, and the atmospheric absorption causes the light energy to decrease with increasing distance. Atmospheric scattering reduces the light energy in the propagation direction and changes the light-intensity distribution of the spot so that it includes light and dark. Atmospheric attenuation commonly influences atmospheric absorption and scattering. The atmospheric transmittance of monochromatic waves can be expressed as [7]

$$\tau(\lambda) = \exp\left(-\int_0^z \alpha(\lambda) dr\right), \quad (6.7)$$

where $\tau(\lambda)$ is the atmospheric transmittance at wavelength λ , z is the transmission distance, and $\alpha(\lambda)$ is the total attenuation coefficient.

$$\alpha(\lambda) = \alpha_m + \alpha_s, \quad (6.8)$$

where α_m is the scattering coefficient and α_s is the absorption coefficient. For a uniform horizontal optical path,

$$\tau(\lambda) = \exp(-\alpha(\lambda)z). \quad (6.9)$$

For an oblique transmission,

$$r(\lambda) = \exp\left(-\sec \phi \int_0^H \alpha(\lambda) dh\right), \quad (6.10)$$

where ϕ is the zenith angle and H is the tilt height.

If the initial luminous flux is I_0 , the luminous flux $I(z)$ after propagation distance z is

$$I(z) = I_0 \times \tau(\lambda). \quad (6.11)$$

b. Estimating transmittance

Using the attenuation coefficient to calculate atmospheric transmittance is only suitable for theoretical calculations. It is not only complex but also inconsistent with the actual situation. This needs to be corrected many times, which causes great trouble in engineering applications.

For the horizontal transmission of light waves, the dominant attenuation of the bottom atmosphere is only Mie scattering. At this time, the atmospheric transmittance can be expressed by an empirical formula related to “visibility.” “Visibility” is a measure of the attenuation of visible light by the atmosphere. During the day, it refers to the longest distance that human eyes can see under the background of a horizontal sky; at night, it refers to the distance at which a medium-intensity unfocused light source can be seen. In meteorology, “visibility” is usually divided into ten levels, according to meteorological conditions, as shown in Table 6.1.

The atmospheric-scattering coefficients under common climatic conditions are listed in Table 6.1. The empirical formula for atmospheric transmittance and visibility is [15]

$$\tau(\lambda) = \exp\left[-\frac{3.912}{V} \left(\frac{\lambda}{0.55}\right)^{-q} \times z\right], \quad (6.12)$$

where V is the visibility in kilometers, z is the transmission distance in kilometers, λ is the wavelength in microns, and q is a constant related to the wavelength. For the

Table 6.1 International visibility levels [7]

Grade	Weather status	Visibility	Scattering coefficient	Grade	Weather status	Visibility (km)	Scattering coefficient
0	Dense fog	<50 m	>78.2	5	Haze	2 4	1.960 0.954
1	Thick fog	50 m 200 m	78.2 19.6	6	Light haze	4 10	0.954 0.391
2	Mid fog	200 m 500 m	19.6 7.82	7	Sunny	10 20	0.391 0.196
3	Light fog	500 m 1 km	7.82 3.91	8	Very sunny	20 50	0.196 0.078
4	Mist	1 km 2 km	3.91 1.96	9	Extremely sunny	>50	0.0141

infrared band, the value of q is

$$q = \begin{cases} 1.6 & \text{larger } V \\ 3 & \text{medium visibility} \\ 0.585 V^{1/3} & V \leq 6 \text{ km} \end{cases} \quad (6.13)$$

6.1.2 Atmospheric-Turbulence Effect

(1) Analysis of a turbulent atmosphere

A laser pulse propagating through the atmosphere not only loses energy to absorption and scattering, but is also affected by atmospheric turbulence. Turbulence produces vortex elements with different refractive indices, owing to the small differences in temperature and density. These vortex elements move rapidly with the wind speed and are constantly generated and annihilated.

The changing spectrum can reach hundreds of Hertz, and the changing spatial scale may be as small as a few millimeters or as large as tens of meters. When the beam passes through these vortex elements with different refractive indices, it experiences atmospheric-turbulence effects, such as beam bending, drift, and expansion distortion, resulting in the flicker and jitter of the received light intensity.

(2) Statistical characteristics of turbulence

In a turbulent atmosphere, the refractive index changes in different places and times. Because its variation appears random, it is necessary to use statistical methods to describe this medium.

a. Atmospheric refractive-index structure constant

The turbulence structure constant C_n^2 is typically used to describe the atmospheric optical turbulence intensity, whereas the actual measurement is the atmospheric refractive-index structure function $D_n(r)$, but only under local uniform isotropy. According to Kolmogorov's turbulence statistical theory, in the inertial sub-interval ($L_0 \gg r \gg l_0$), the relationship between $D_n(r)$ and C_n^2 is as follows:

$$D_n(r) = C_n^2 r^{\frac{2}{3}} (l_0 \ll r \ll L_0), \quad (6.14)$$

where r is the distance between two points of statistical-turbulence characteristics, and l_0 and L_0 are the inner and outer turbulence scales, respectively.

C_n^2 , also known as the atmospheric refractive-index structure constant, is one of the basic parameters of atmospheric optics and an important evaluation parameter for turbulence intensity. The smaller C_n^2 is, the weaker the air flow is. It varies greatly with the geographical location, altitude, meteorological conditions, season, day, night, and other conditions. Figures 6.2 and 6.3 show typical examples of C_n^2 changes in a 24-h day and night, respectively.

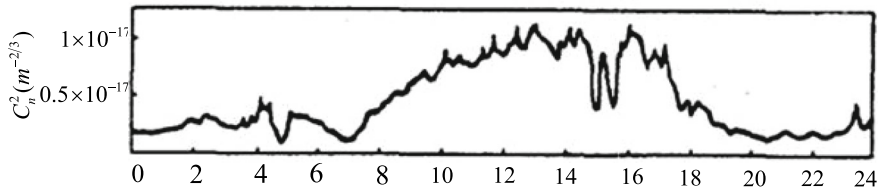
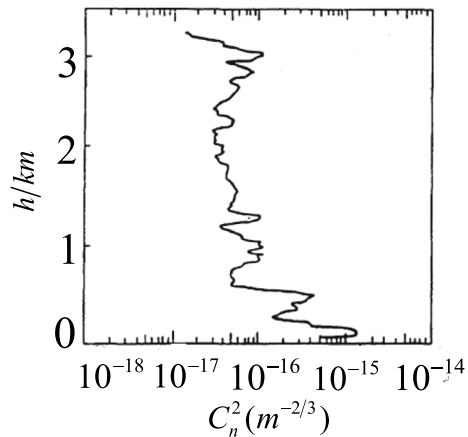


Fig. 6.2 Variation of the refractive-index structure constant on a sunny day for 24 h [17, 18]

Fig. 6.3 Variation of the refractive-index structure constant with altitude [17, 18]



The general conditions of turbulence variation are as follows: Summer is stronger than winter; sunny days are stronger than cloudy days. It is strongest around noon (10:00–15:00) and weak at other times. The refractive-index structure constant also exhibits the same variation patterns. As shown in Fig. 6.2, the turbulence is very weak at night and in the morning. With the sunrise, C_n^2 increases rapidly, and does not decrease significantly until the sun sets. In addition, the turbulence intensity decreases with an increase in altitude and C_n^2 jumps with the altitude. The typical value of C_n^2 is 10^{-17} – $10^{-12} \text{ m}^{-2/3}$.

Generally, the values near the ground are divided into three categories:

$$\begin{cases} C_n^2 \geq 10^{-12} \text{ m}^{-2/3} & \text{strong turbulence} \\ C_n^2 \approx 10^{-14} \text{ m}^{-2/3} & \text{moderate turbulent} \\ C_n^2 \leq 10^{-16} \text{ m}^{-2/3} & \text{weak turbulence} \end{cases}$$

b. Refractive-index fluctuations in power-spectral density

The power-spectrum characteristics of refractive-index fluctuation reflect the statistical characteristics of refractive-index field fluctuation in turbulent media to a certain extent. Atmospheric refractive-index fluctuation can be regarded as an incremental stationary random process. According to the incremental random process, the structural function of the refractive-index fluctuation is

$$D_n(\vec{r}_1, \vec{r}_2) = E\left\{[n_1(\vec{r}_1) - n_1(\vec{r}_2)]^2\right\} = D_n(\vec{r}_1 - \vec{r}_2) = D_n(\vec{\rho}). \quad (6.15)$$

The correlation function is

$$B_n(\vec{r}_1, \vec{r}_2) = E\{n_1(\vec{r}_1)n_2(\vec{r}_2)\} = B_n(\vec{r}_1 - \vec{r}_2) = B_n(\vec{\rho}). \quad (6.16)$$

The structural function is a basic feature of the incremental stationary stochastic process and plays an important role. It describes the fluctuation intensity of $\xi(t)$ in period τ . The correlation and structure functions have the following relationships:

$$D_n(\vec{\rho}) = 2[\text{Var}(n_1) - B_n(\vec{\rho})] = -2B_n(\vec{\rho}). \quad (6.17)$$

The spectral density, owing to refractive-index fluctuation, is [13]

$$\Phi_n(K) = (2\pi)^{-3} \int_V B_n(\vec{\rho}) - \exp(i\vec{K} \cdot \vec{\rho}) dV. \quad (6.18)$$

Thus, we can obtain

$$D_n(\vec{\rho}) = (2\pi)^{-3} \int_V \Phi_n(\vec{K}) [1 - \exp(i\vec{K} \cdot \vec{\rho})] dV, \quad (6.19)$$

where dV is the scattering volume element for $\vec{\rho}$ and V is the scattering volume element for $\vec{\rho}$. The “ $-11/3$ ” law of the refractive-index spectral-density function $\Phi_n(K)$ can be derived from Eq. (6.14):

$$\Phi_n(K) = 0.033 C_n^2 K^{-\frac{11}{3}}. \quad (6.20)$$

Based on atmospheric-turbulence characteristics, Karman deduced that the spectral density of the refractive-index fluctuation is [18]

$$\Phi_n(K) = 0.033 C_n^2 \exp \left[- \left(\frac{K l_0}{2\pi} \right)^2 \right] \left[K^2 + \left(\frac{2\pi}{L_0} \right)^2 \right]^{-11/6}. \quad (6.21)$$

The spectrum expansion of the refractive-index structure function is [18]

$$D(\rho, z) = 8\pi^2 k^2 z \int_0^\infty [1 - J_0(K\rho)] \Phi_n(K) K dK, \quad (6.22)$$

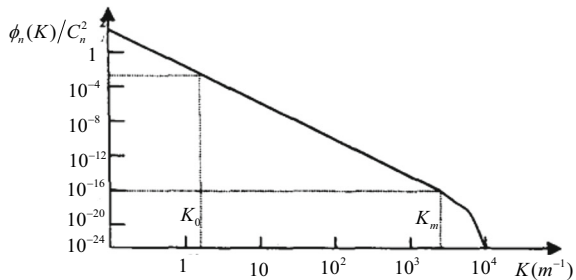
where k is the Boltzmann constant, K is the spatial wave number of the turbulence, and J_0 is the zero-order Bessel function. Figure 6.4 shows the most basic turbulence model, which is the commonly used refractive-index power spectrum.

(3) Influence of turbulence on laser transmissions

The influence degree and form of atmospheric turbulence on beam characteristics are closely related to the beam diameter D and turbulence scale L , which can be roughly divided into three cases:

1. $d \ll L$. When the beam diameter is much smaller than the turbulence scale, the turbulence randomly deflects the beam and causes it to drift at the receiver.
2. $d \approx L$. When the turbulence scale is approximately equal to the beam diameter, the turbulence randomly deflects the beam section, resulting in the fluctuation of the arrival angle and image-point jitter on the focal plane at the receiving end.
3. The more common situation is $d \gg L$; that is, the beam diameter is significantly larger than the turbulence scale. At this time, the beam section contains

Fig. 6.4 Refractive-index fluctuation power spectrum [18]



many small turbulent vortices, which diffract a small part of the irradiated beam, resulting in a random distribution of the beam intensity and phase in space and time, coherence degradation, and beam-area expansion, resulting in a fluctuation of the light intensity at the receiving end. In addition, the overall received light intensity is attenuated.

In practice, the temperature-difference disturbance continuously mixes with the atmosphere and produces many unpredictable turbulence elements of various scales. These turbulence elements work together to strengthen the light-intensity fluctuation at the receiving end. The light-intensity fluctuation is also related to the wind speed and meteorological conditions at that time. Therefore, it is difficult to detect and observe atmospheric turbulence, which poses a significant obstacle to stable communication in optical-communication systems. At present, adaptive optics can better solve this problem, but more experiments are still needed to explore its change scale and law.

a. Intensity fluctuation (atmospheric scintillation)

The atmospheric-scintillation effect (intensity scintillation) occurs when the beam diameter is much larger than the turbulence scale. The beam section contains multiple turbulent vortices, and each vortex scatters and diffracts independently, compared with the part of the beam on it. This results in fluctuation of the light intensity; that is, atmospheric scintillation.

In laser-communication systems, atmospheric scintillation can cause random fluctuations in the detection current of the receiver, resulting in an increase in the noise of the detection system. Figure 6.5 shows a schematic diagram of a beam propagating through a turbulent atmosphere.

The intensity fluctuation of a laser beam is usually expressed by the logarithmic-intensity fluctuation variance [19, 20]. To satisfy the Kolmogorov spectrum, the variance I of the logarithmic-intensity fluctuation is

$$\sigma_{\ln}^2 I = AC_n^2 k^{7/6} L^{11/6}, \quad (6.23)$$

where L is the horizontal-transmission distance; A is a constant, which is 0.496 and 1.23 for spherical and plane waves, respectively; wave number $k = 2\pi/\lambda$, and λ is the laser wavelength.

According to Eq. (6.23), I is directly proportional to the transmission distance to the power of 11/6, and the wave number to the power of 7/6. The scintillation of a

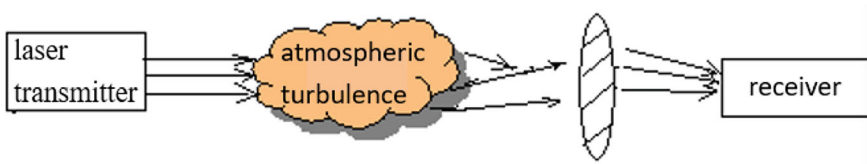


Fig. 6.5 Schematic diagram of a laser beam propagating through a turbulent atmosphere

plane wave is larger than that of a spherical wave. The scintillation size is directly proportional to the turbulence intensity; the shorter the wavelength, the longer the transmission distance and the stronger the flicker. However, when the turbulence intensity is large and the transmission distance is long, the change in I is much more complex than that in this formula.

When the laser beam is horizontally transmitted near the ground, I can reach more than 1 over a short distance. However, theory and experiments show that after reaching 1–2, it no longer increases with the increase in turbulence intensity and transmission distance, but may decrease. This phenomenon is known as the scintillation-saturation effect. When a visible-band laser passes upward or downward through the atmosphere, the value is approximately 0.02. Hence, this intensity fluctuation will not have a significant impact on laser applications.

In addition, scintillation is also related to weather conditions. Generally, scintillation is strong on sunny days and weak on cloudy days, which is the opposite of atmospheric attenuation, which is small on sunny days and large on cloudy days.

b. Beam drift and image-point jitter

When a laser passes through turbulent atmosphere, random fluctuations in the propagation direction cause the beam to deviate from the expected position. This effect is called beam drift and is typically measured by the drift angle or drift amplitude. Simultaneously with the drift, the beam's angle of arrival on the receiving plane also fluctuates randomly, owing to the influence of turbulence. If observed with a telescope, the image point cannot be seen in the same position on the focal plane. This phenomenon is called image-point jitter, and the magnitude of the jitter is measured by the variance of the angle-of-arrival fluctuation.

The drift angle is the angle at which the center of the spot deviates from its average position, and the angle of arrival is the divergence angle when the beam is incident on the receiving surface. It can be proven that the relationship between the drift-angle fluctuation variance σ_α^2 and arrival-angle fluctuation variance σ_θ^2 is

$$\sigma_\alpha^2 = \frac{1}{3}\sigma_\theta^2. \quad (6.24)$$

According to Tatarski's theory, the variance in the angle-of-arrival fluctuation, when the turbulence is not very strong, can be expressed as

$$\sigma_\theta^2 = 2.92C_n^2 z \rho^{-1/3}, \quad (6.25)$$

where ρ is the aperture of the receiving optical device. The probability of the beam drift and image-point jitter follows a normal distribution [17]. Generally, the fluctuation of the drift angle and arrival angle is less than 50 μrad .

Figure 6.6 shows the diurnal variation in the beam drift. The refractive-index structure constant is also drawn for comparison. It can be observed that the optical drift angle and refractive-index structure constant change with time. When the refractive-index structure constant increases, the drift angle also increases and vice versa.

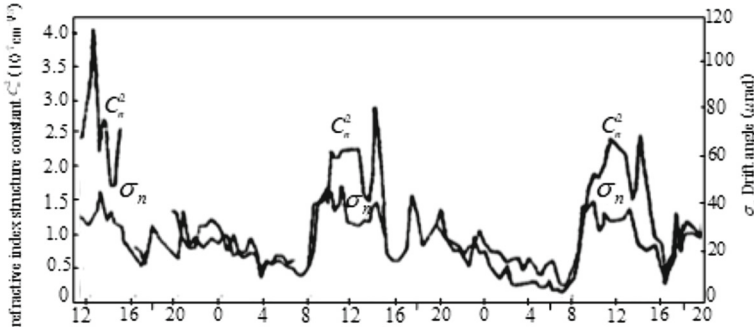


Fig. 6.6 Diurnal beam-drift variation [17]

Because the refractive-index structure constant is a physical quantity that represents the turbulence intensity, the beam drift is closely related to the turbulence intensity. When the turbulence is strong, the drift angle is large, whereas when the turbulence is weak, the drift angle is small.

c. Signal-to-noise ratio influence

Owing to the influence of atmospheric turbulence on the transmitted laser beam, the optical power received by the receiver varies randomly. The signal-to-noise ratio (SNR) calculation of a direct-detection system should include the power-variance factor Δp_r caused by the atmospheric-turbulence effect.

Assuming that the diameter of the receiving antenna is d , the signal field is a Gaussian field, and r_0 is the coherence length on the transverse plane of the transmitted beam, the signal-to-noise ratio under the quantum-limit condition is

$$SNR = \frac{1}{\left| \frac{\alpha \bar{P}_r}{2F \cdot B} \right| + \Delta P_r}, \quad (6.26)$$

where $\Delta p_r = \frac{2}{d^2} \int_0^1 \exp \left| - \left| \frac{ud}{2r_0} \right|^2 \right| \cdot \left| \frac{1}{\cos u} - u\sqrt{1-u^2} \right| u du$.

It can be observed from the above formula that the atmospheric-turbulence effect reduces the signal-to-noise ratio of a direct-detection system.

6.1.3 Principle of an Experimental Measurement System [21–23]

The working principle of a laser far-field real-time spot-measurement system is as follows: After the signal sent by a semiconductor laser is collimated and transmitted by the transmitting antenna, it passes through a random atmospheric channel to

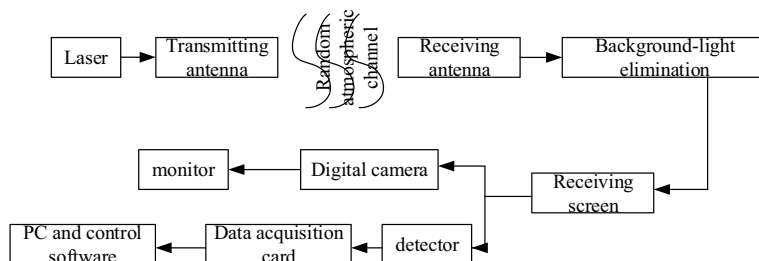


Fig. 6.7 Schematic diagram of a real-time spot-measurement system [21]

eliminate the interference of background light. After being received by the receiving antenna, it shines on a receiving screen at the receiving end.

The output video signal is recorded in real time with a digital camera, and the output signal is processed and input into a computer to display the dynamic change of the spot in two-dimensional real time. Simultaneously, one end of the detector is aligned with the light-spot image, and the other end is connected to a data-acquisition card. The collected data are transmitted to a computer through serial-port communication for calculation, storage, and analysis. The working principle of the real-time spot-measurement system is shown in Fig. 6.7.

After the laser-power meter converts the optical signal into an electrical signal for output, the signal-conditioning circuit amplifies the output signal. In addition, to ensure the accuracy of the analog/digital conversion, a sampling/holding circuit is connected between the analog signal source and the ADC (Analog to Digital Converter), and an A/D conversion circuit performs an analog/digital conversion under the control of a single-chip microcomputer, collecting power data that changes linearly with the voltage over time.

Through serial communication between the single-chip microcomputer and the upper computer, the system can set the acquisition time, query and delete historical data, and monitor the power in real time. The principle block diagram of the data-acquisition system is shown in Fig. 6.8.

6.2 Analysis of the Measurement Results of a Light-Intensity Experiment [21]

We conducted several field experiments under typical weather conditions. From these experimental results, we can study the influence of the atmosphere on a laser transmission in an atmospheric channel and intuitively observe the attenuation and turbulence effects of the atmosphere on the laser transmission. This makes our research on atmospheric channels easier, more intuitive, and more accurate.

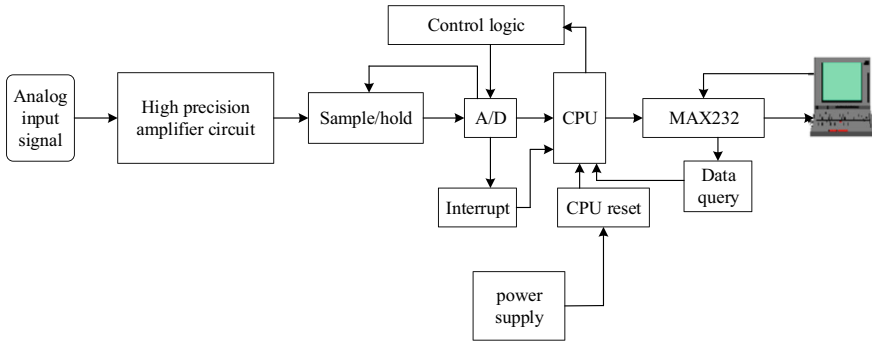


Fig. 6.8 Schematic block diagram of data-acquisition card [21]

6.2.1 Daily Variation of the Light-Intensity Scintillation

Atmospheric turbulence often occurs on sunny days, and when the sky clears after cloudy and rainy days. In the experimental measurements, we observed strong light-intensity flicker, especially after sunrise in the morning, and in the afternoon. On sunny days with good visibility, we believe that this is mainly caused by atmospheric turbulence. Atmospheric turbulence makes the channel very unstable. The power-monitoring value shows that the light intensity received by the detector varies from large to small, the frequency changes rapidly, and the fluctuation range is large.

(1) Variation law

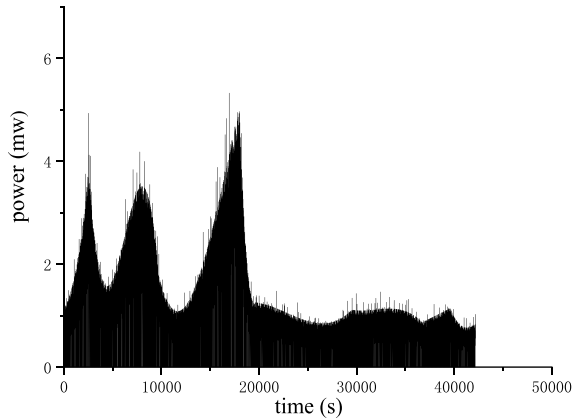
The daily variation law of the statistical characteristics of the light-intensity scintillation in each month is roughly the same, with some differences. We elaborate on their characteristics, based on the situation during one day in July.

Figure 6.9 shows the daily variation in light-intensity flicker from 8:48 to 22:00 on July 2. It was a sunny day with good visibility and a temperature of 33 °C. One point was collected every second, for a total of 47,520 points. We observed that the fluctuation intensity began to rise after the sun came out and then began to decline after reaching a peak. It was the strongest at approximately 13:30 and then gradually began to decline. It increased slightly at sunset, decreased significantly in the evening, and then increased slightly at night.

The general feature is that the fluctuation intensity at night was slightly lower than that during the day. The flicker was strongest from 13:00 to 15:00 in the afternoon, small after sunrise and before sunset, and smallest in the evening. From 21:00 to 22:00 in the evening, it showed an upward trend. Except for a high fluctuation intensity around noon, weak fluctuation conditions continued throughout the day. This is consistent with the turbulence theory that atmospheric turbulence is mainly caused by atmospheric-temperature-difference disturbances due to solar-energy decay.

According to atmospheric-turbulence theory, the ground temperature rises gradually after the sun rises in the daytime, and the difference between the air temperature and ground temperature increases continuously. At noon, this difference reaches a

Fig. 6.9 Daily variation of light-intensity scintillation [21]



maximum value. The atmospheric turbulence caused by the temperature difference also increases gradually, and is the strongest before and after noon. The random-variation range of the refractive index on the channel increases, and the variation frequency is fast; thus, the light-intensity fluctuation gradually increases from sunrise, with a maximum and the most intense flicker at noon.

At sunrise or sunset, the ground temperature is almost the same as the air temperature, and the atmospheric turbulence is weak; therefore, the flicker is small. At night, the ground temperature gradually decreases. At this time, the ground temperature is lower than the air temperature, which increases the turbulence; thus, the flicker exhibits an upward trend.

The experimental system was used to continuously measure for 12 h, collecting one point per second, and observe the diurnal variation of the light-intensity flicker. This is consistent with the results of atmospheric-turbulence theory.

(2) Probability-distribution characteristics of light-intensity fluctuation

At present, the probability-distribution models of light-intensity scintillation include the lognormal, K , M , and Rayleigh distributions. The results show that under weak-fluctuation conditions, the probability density of the light-intensity fluctuation obeys a lognormal distribution, and under a strong fluctuation, it follows an exponential distribution.

6.2.2 Impact of Different Weather Conditions on an Optical-Communication Link

(1) Smog, fog, and suspended-particle weather

Suspended particles in the atmosphere, such as dust, smoke, and industrial pollutants, mainly originate from human activities, volcanic eruptions, plant secretions, and cosmic dust. There are significant differences between regions and environments.

The “big” particles, with a diameter ranging from 0.1 to 10 μm , have a great impact on light propagation. Sometimes, such a phenomenon occurs in the observatory in Xi'an, China: It is sunny at high altitudes, but the atmosphere near the ground is gray, and the air is filled with suspended particles, such as dust and smoke, which affect the optical communication link. They scatter the optical signal, resulting in a continuous attenuation of energy and continuous loss of carried data. In serious cases, the receiver cannot detect the signal, and the link communication is interrupted.

When the water–vapor content in the atmosphere reaches or exceeds the saturation state, it condenses into water droplets on the air-soluble colloidal particles. Condensed water droplets float in the air near the ground, reducing the visibility distance. When the visibility distance is less than 1 km, it is called fog. When the visibility distance is between 1 and 10 km, it is called light fog or haze. The attenuation of light energy by fog is significant.

In the experiment, it was observed that fog had a significant impact on the optical-communication link. When the visibility was less than 1 km, the power-monitoring value dropped to 0.1–0.3 (the power-indication value is generally 1–5 in sunny weather, up to more than six). When the visibility was approximately 500 m, the signal could not be detected at the receiving end and the power-monitoring value was 0.

Figure 6.10 shows foggy weather with poor air quality and large amounts of smoke. The temperature was 23 °C. The measurement time was from 13:30 to 13:46, at 20 points per second, for a total of 28,560 points.

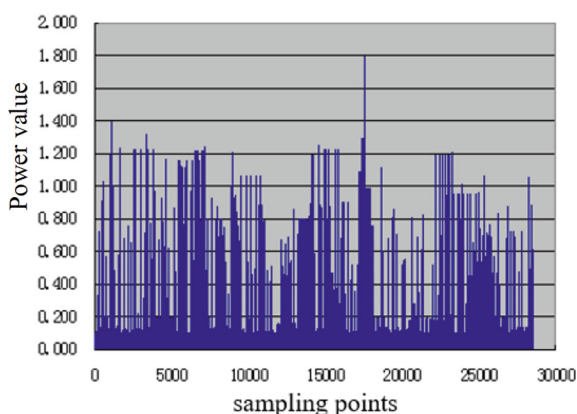


Fig. 6.10 Fog sampling data [21]

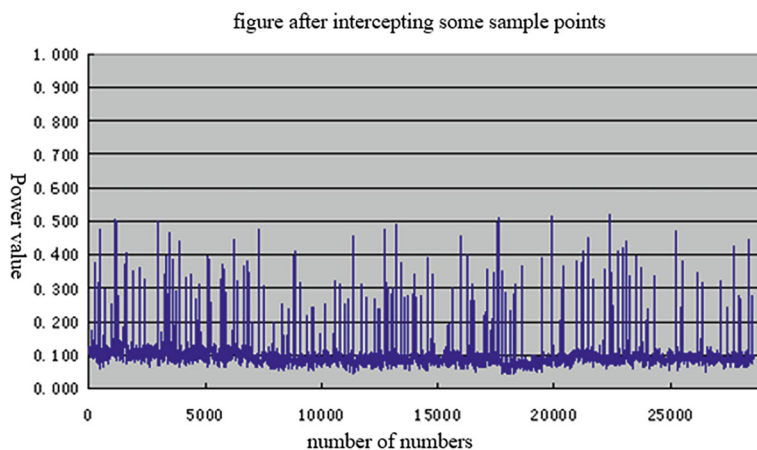


Fig. 6.11 Sampling data on a cloudy day [21]

(2) Cloudy days

On cloudy days, the background-light interference was small, and had little effect on the spot-intensity fluctuation compared to sunny days. The power-monitoring value showed that it was approximately 0.1 without obvious changes. Figure 6.11 shows the measurement results on a cloudy day. The temperature was 22 °C, the time was 16:41–17:07, at 20 points per second, for a total of 30,200 points.

(3) Light rain

When a water drop in the air reaches a certain radius, it no longer floats, but falls to the ground at a certain rate to become rain. Daily precipitation greater than 25 mm is called heavy rain; daily precipitation greater than 50 mm is called rainstorm; and daily precipitation greater than 100 mm is called heavy rainstorm.

Generally, the scattering attenuation of light due to raindrops is not very large, and the flicker variance is small. This is because rain not only reduces the ground temperature and temperature difference, but also reduces the uneven components in the air. During the experiment, it was observed that the power-monitoring value was below 0.1, which was smaller than that on cloudy days, and there was no obvious change. Moderate rain and light rain have little impact on the optical-communication link.

Figure 6.12a shows the data obtained when the air temperature was 23 °C, the measurement time was 13:48–14:10, at 20 points collected per second, and 2000 sample points intercepted and Collected. Figure 6.12b shows the graph obtained when the temperature was 22 °C and the measurement time was 16:41–17:31. We collected 20 points per second and intercepted 5000 sample points.

(4) Heavy rain

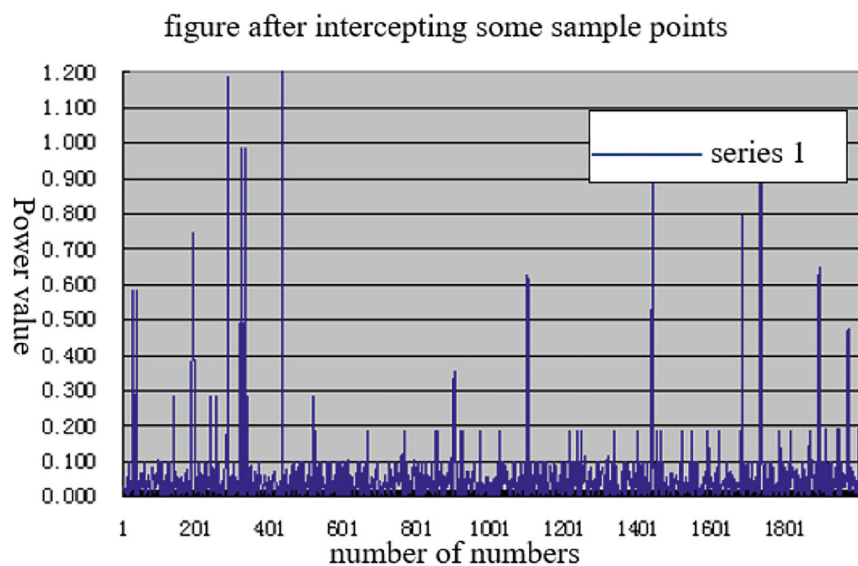
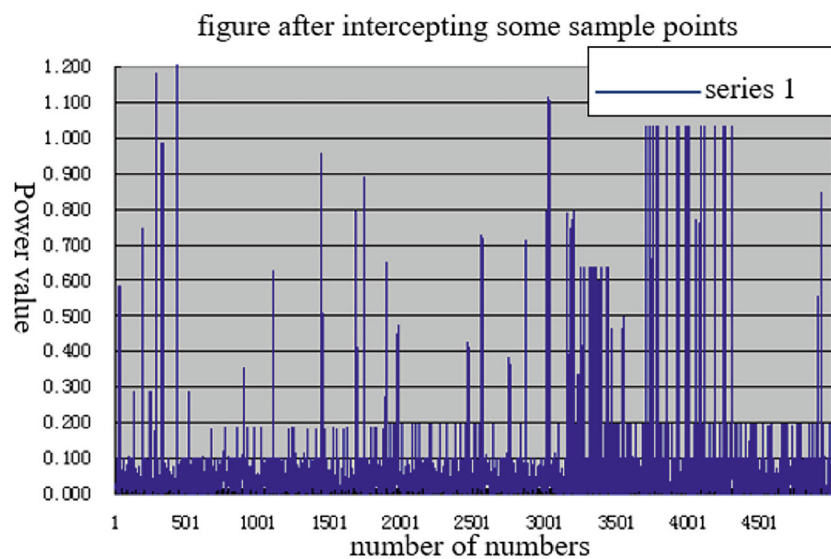
In the case of heavy rain, the nature of the atmospheric channel becomes complex and a sudden change in weather causes a sudden change in the optical power of the detector. It was observed in the experiment that the power value changed significantly during heavy rain. It can be seen that the change frequency of the indicated value on the power meter was rapid, indicating that it had a great impact on the optical communication at this time, and many times the power value appears as 0.

Figure 6.13a shows the graph obtained at a temperature of 20 °C and a measurement time of 12:15–12:27 after collecting 20 points per second and intercepting sampling points. At this time, there was sudden heavy rain and fog, and the indicated value of the power meter changed between 0 and 0.2.

Figure 6.13b shows the data collected when the temperature was 21 °C and the measurement time was 12:30–12:40. At this time, the weather was steady, moderate to heavy rain

6.2.3 Spot-Jitter Analysis

When atmospheric-turbulence motion is obvious, the scintillation of the beam intensity can be observed with the naked eye. In this experiment, the received light intensity was measured. Figure 6.14 shows a typical measurement curve of the spot and received-light power jitter. Figure 6.14a shows an image of the spot. The jitter of the light intensity is shown in Fig. 6.14b. In addition, the dynamic change in the spot recorded by the digital camera through the atmospheric channel can reveal the jitter of the image point.

(a) Mapping of sampling data on light rainy days ^[21](b) Mapping of sampling data on light to moderate rainy days ^[21]**Fig. 6.12** Light-power attenuation in light rain and light-to-moderate rain ^[21]

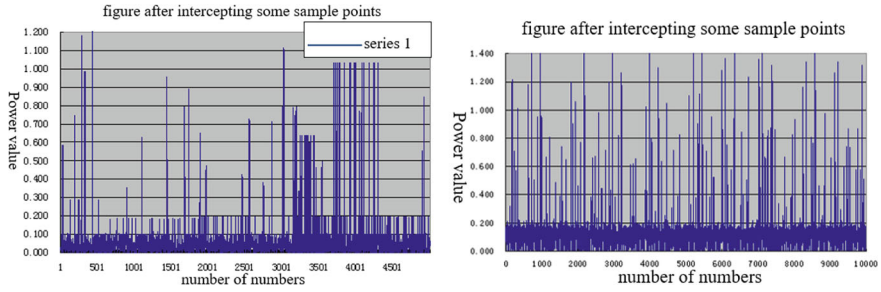


Fig. 6.13 Optical power attenuation in rainy weather under different conditions [21]. **a** Sudden heavy rain and fog. **b** Moderate to heavy rain

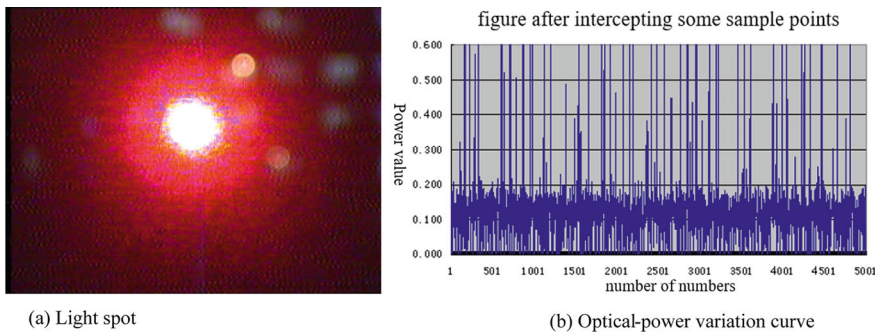


Fig. 6.14 **a** Light spot. **b** Measurement curve of the received-light power jitter [21]

6.3 Bit-Error Rate Simulation and Measurement-Result Analysis

6.3.1 Numerical Simulation of the Bit-Error Rate Under Different Weather Conditions

(1) RS codes

We calculated the variation curves of the bit-error rate with the signal-to-noise ratio of the Reed–Solomon (RS) (15, 9) and RS (30, 10) codes under different weather conditions, as shown in Figs. 6.15, 6.16, 6.17 and 6.18. With an increase in the SNR, the performance of the RS (30, 10) code was better than that of the RS (15, 9) code.

(2) Turbo codes

We calculated the curve of the bit-error rate versus the signal-to-noise ratio of turbo codes with a code rate of 1/3 and interleaving length of 64 under different weather conditions [11], as shown in Figs. 6.19 and 6.20.

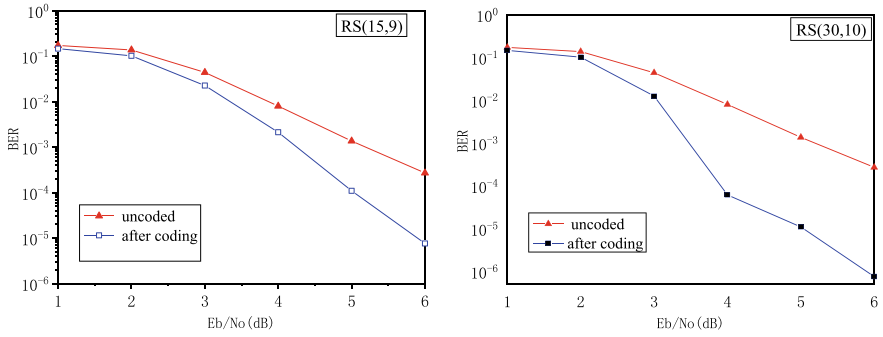


Fig. 6.15 Error characteristics of RS (15, 9) and RS (30, 10) codes in sunny weather [23]

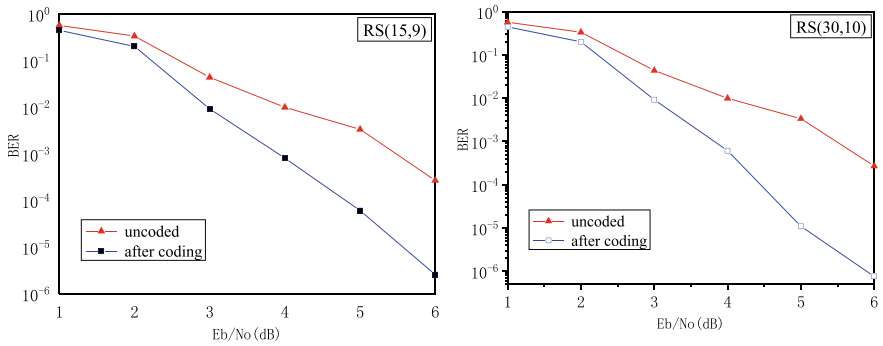


Fig. 6.16 Error characteristics of RS (15, 9) and RS (30, 10) codes in cloudy weather [23]

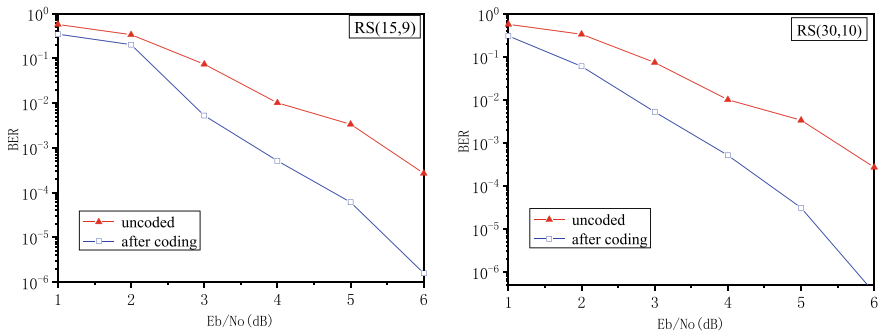


Fig. 6.17 Error characteristics of RS (15, 9) and RS (30, 10) codes in rainy weather [23]

(3) LDPC codes

Based on the actual measured atmospheric-channel attenuation, we calculated the performance of atmospheric laser communication using low-density parity-check

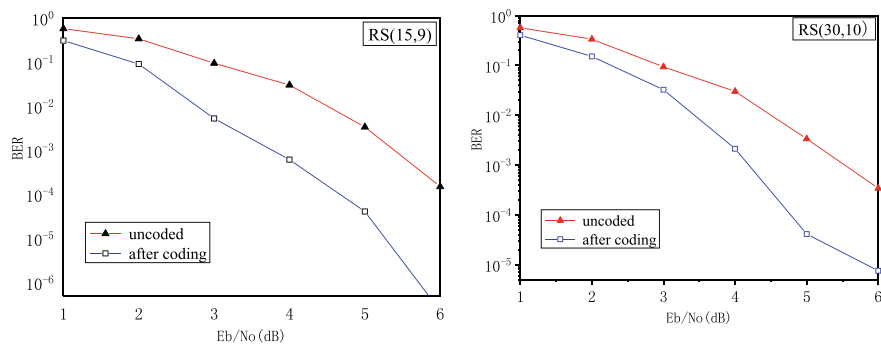


Fig. 6.18 Error characteristics of RS (15, 9) and RS (30, 10) codes in foggy weather [23]

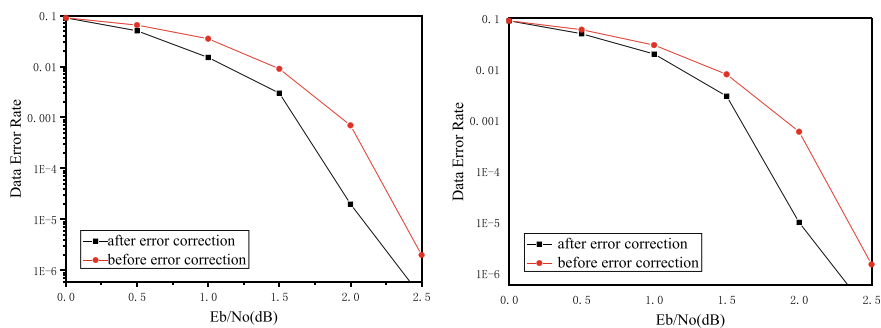


Fig. 6.19 System BER performance curves before and after the introduction of turbo codes. **a** Error correction on sunny days; **b** error correction on cloudy days

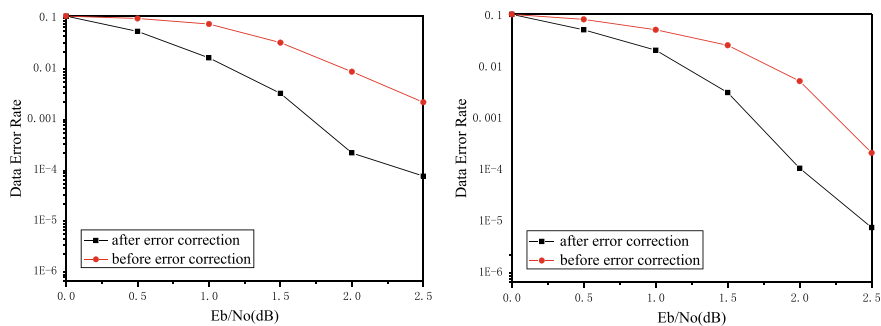


Fig. 6.20 System BER performance curve before and after the introduction of turbo codes. **a** Error correction on rainy days; **b** error correction on foggy days

(LDPC)-code error correction under different weather conditions [9], as shown in Fig. 6.21.

6.3.2 Bit-Error Rate Measurement

The experiment was conducted from September 2006 to October 2007, during which time, various weather conditions were experienced. The experiment was conducted between two buildings in inclined transmission mode. The measurement results are presented in Table 6.2. The system bit-error rates were measured under different weather conditions and error-correction methods. The RS error-correction code adopts the RS (15, 9) code and the cyclic shortened RS (30, 10) code. The LDPC code had a code rate of 1/2 and a code length of 128. The turbo code had a 1/3 bit rate, and the interleaving length was 64.

The experimental results show that the wireless laser-communication system can work normally in various weather conditions; however, when it encounters severe weather, such as heavy fog, heavy rain, and heavy snow, the bit-error rate of the system increases. The bit error rate has reached 10^{-3} or nearly 10^{-2} . However, using atmospheric channel-coding technology, the target error rate after error correction can be achieved 10^{-5} . Specifically, the turbo and RS (30, 10) error-correction codes have relatively good error-correction performance for optical communication systems, and their improvement effect on the system performance is obvious. After error correction, the error rate of the system is all below 10^{-5} .

In the following section, the influence of atmospheric channels on atmospheric laser communication, owing to meteorological conditions, can be addressed. In general, in light rainy weather, the average bit-error rate of the system is maintained at 10^{-4} . The performance of the atmospheric laser-communication system was improved using RS (15, 9) and LDPC codes, and the bit-error rate of the system after the improvement reached 10^{-6} . If the average bit-error rate of the system before error correction is less than 10^{-7} , the bit-error rate of the system after error correction is even higher on sunny and cloudy days.

The distribution of errors occurring in 16 time slots in a frame is basically uniform, and no time slot is more dominant. We used data from July 4 and 5, and August 9, 2007, as samples. The number of symbol errors was 69,120; that is, the number of errors in a PPM frame was 69,120. The number of errors in the first eight time slots was 32,486, accounting for 47%; the number of errors in the last eight time slots was 36,634, accounting for 53%. The number of errors between adjacent time slots was 4838, accounting for 7%; the number of errors between two time slots was 6221, accounting for 9%; and the number of errors between more than two time slots was 58,061, accounting for 85%.

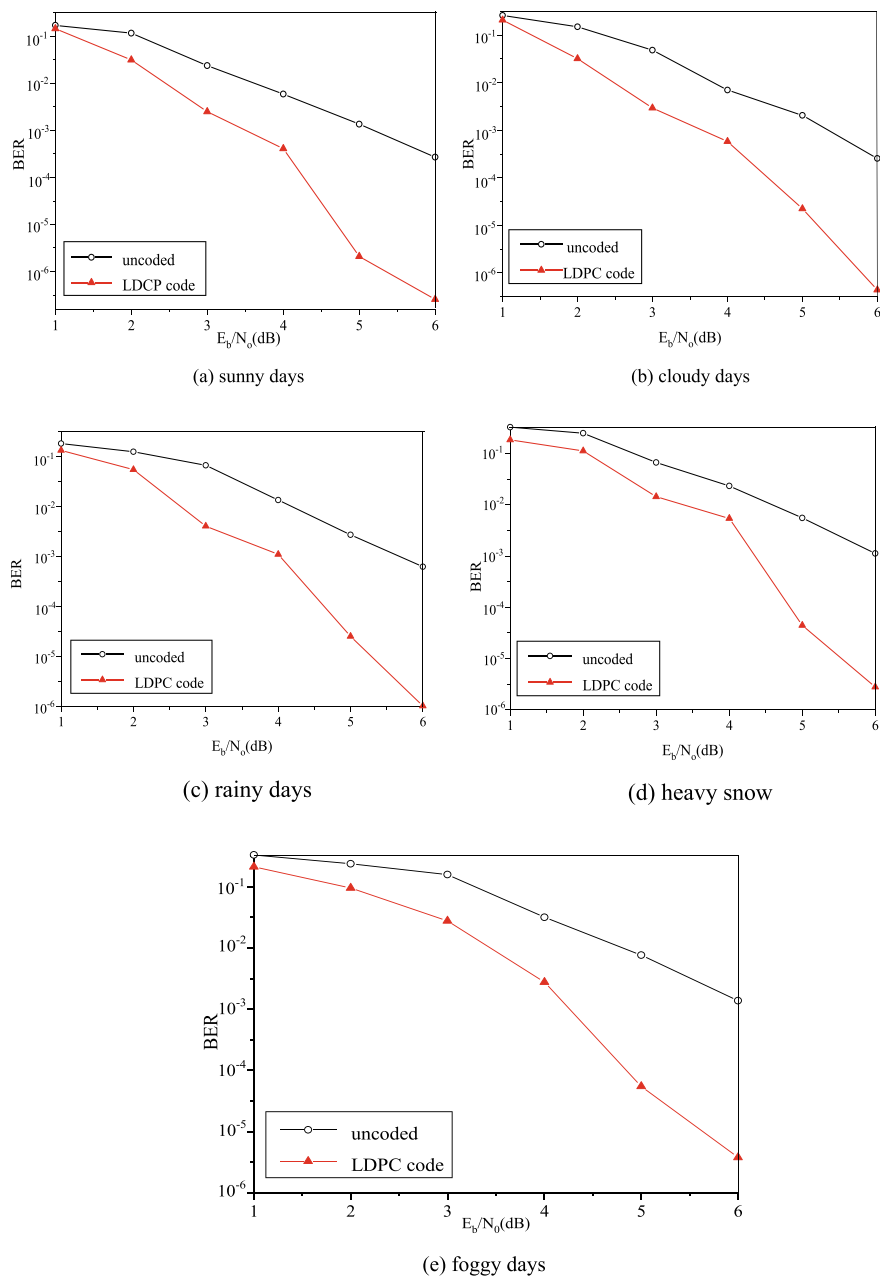


Fig. 6.21 Variation of the bit-error rate before and after LDPC code error correction in different weather conditions

Table 6.2 (a) Bit-error rate under foggy weather conditions; (b) Bit-error rate under rainy weather conditions; (c) Bit-error rate under snowy weather conditions

(a)

Date	Weather condition	Visibility	BER before error correction	Error-correction mode			
				LDPC	Turbo	RS (15, 9)	RS (30, 10)
2006-10-26	Dense fog	200 m	3.41×10^{-3}	5.74×10^{-5}		5.10×10^{-5}	
2006-10-27	Dense fog	200 m	3.35×10^{-3}	2.96×10^{-5}		2.72×10^{-5}	
2007-12-8	Rain and fog	150 m	7.3×10^{-3}		5.4×10^{-5}		3.2×10^{-5}
2007-12-9	Dense fog	200 m	2.6×10^{-3}	4.4×10^{-5}			7.6×10^{-6}
2007-12-10	Dense fog	400 m	1.3×10^{-3}			8.4×10^{-6}	2.5×10^{-6}
2006-9-10	Sunny	15 km	$\leq \times 10^{-7}$	$\leq \times 10^{-9}$			
	Overcast	3 km	$\leq \times 10^{-6}$	$< \times 10^{-9}$			

(b)

Date	Weather condition	Rainfall (mm)	BER before error correction	Error-correction mode			
				LDPC	Turbo	RS (15, 9)	RS (30, 10)
2007-4-23	Light rain	0.2	2.2×10^{-4}	3.3×10^{-6}		3.0×10^{-6}	
2007-7-4	Heavy rain	19.8	6.8×10^{-4}		4.0×10^{-6}		2.4×10^{-7}
2007-6-17	Moderate rain	27.5	3.5×10^{-4}		8.4×10^{-7}		6.8×10^{-7}
2007-7-5	Rainstorm	34.1	2.4×10^{-3}	4.2×10^{-5}	8.8×10^{-6}	3.2×10^{-5}	
2007-8-9	Heavy rain	28.6	7.7×10^{-4}		2.2×10^{-6}		1.5×10^{-6}
2007-8-26	Heavy rainstorm	76.9	8.2×10^{-3}			1.04×10^{-5}	
2007-8-30	Light to moderate rain	21.4	4.4×10^{-4}			5.0×10^{-6}	6.4×10^{-7}
2007-8-31	Moderate to light rain	2.4	3.8×10^{-4}		1.06×10^{-6}	4.8×10^{-6}	
2007-9-3	Moderate to light rain	10.1	4.0×10^{-4}			5.4×10^{-6}	8.6×10^{-7}

(continued)

Table 6.2 (continued)

(b)							
Date	Weather condition	Rainfall (mm)	BER before error correction	Error-correction mode			
				LDPC	Turbo	RS (15, 9)	RS (30, 10)
2007-9-7	Moderate to heavy rain	33.5	5.8×10^{-4}		3.4×10^{-6}		
2007-9-27	Light rain	7.0	2.6×10^{-4}	4.2×10^{-6}	8.3×10^{-7}		
2007-9-28	Light rain	8.7	3.2×10^{-4}			4.0×10^{-6}	
2007-9-29	Light rain	3.0	3.5×10^{-4}	4.3×10^{-6}		3.8×10^{-6}	
2007-10-8	Drizzle	0.2	7.6×10^{-5}			5.3×10^{-7}	
2007-10-9	Light rain	1.6	2.0×10^{-4}	3.2×10^{-6}	7.8×10^{-7}	3.0×10^{-6}	7.2×10^{-7}
2007-10-10	Light rain	6.6	2.6×10^{-4}	3.5×10^{-6}		3.1×10^{-6}	
2007-10-11	Light to moderate rain	12.0	4.8×10^{-4}	6.6×10^{-6}	1.8×10^{-6}		
2007-10-12	Light to moderate rain	24.6	4.0×10^{-4}			5.4×10^{-6}	8.8×10^{-7}
2007-10-27	Heavy rain	20.3	7.5×10^{-4}		2.4×10^{-6}		1.8×10^{-6}
2007-5-13	Moderate rain	13.6	5.0×10^{-4}	6.6×10^{-6}	7.5×10^{-7}	5.2×10^{-6}	
2007-03-15	Moderate rain	10.3	3.3×10^{-4}	5.0×10^{-6}		6.0×10^{-6}	
2007-7-2	Moderate rain	11.9	2.8×10^{-4}		5.4×10^{-7}		4.5×10^{-7}
2007-5-23	Light to moderate rain	9.2	4.7×10^{-4}			5.2×10^{-6}	6.1×10^{-7}
2007-5-24	Cloudy and rainy	0.8	1.5×10^{-4}	1.8×10^{-6}		2.0×10^{-6}	
2007-7-4	Moderate to heavy rain	19.8	6.2×10^{-4}	8.4×10^{-6}		8.2×10^{-6}	
2007-5-23	Moderate rain	12.2	4.5×10^{-4}	5.4×10^{-6}	7.3×10^{-6}		

(continued)

Table 6.2 (continued)

(c)							
Date	Weather condition	Snowfall (mm)	BER before error correction	Error-correction mode			
				LDPC	Turbo	RS (15, 9)	RS (30, 10)
2007-1-3	Moderate snow	1.9	4.6×10^{-3}	7.8×10^{-5}		8.4×10^{-5}	
2008-1-11	Moderate snow	2.5	3.4×10^{-4}		2.1×10^{-6}		1.2×10^{-6}
2008-1-12	Heavy snow	5.6	6.2×10^{-3}	8.8×10^{-5}			3.6×10^{-5}
2008-1-18	Heavy snow	6.4	5.5×10^{-3}		3.8×10^{-5}		2.5×10^{-5}
2008-1-20	Heavy snow	6.0	4.4×10^{-3}		1.6×10^{-5}		9.2×10^{-6}
2008-1-21	Heavy snow	4.8	4.5×10^{-3}	6.3×10^{-5}		5.7×10^{-5}	

6.3.3 Error-Correction Performance Analysis

(1) RS (15, 9) code

According to the measurement results in Table 6.2, when using the RS (15, 9) code to correct system errors, in the case of good weather, such as sunny and cloudy days, the bit-error rate of the system before error correction was maintained 10^{-7} , and the bit-error rate after error correction was improved, it less than 10^{-9} . As the bit-error rate increased, the error-correction performance decreased.

In heavy rain, the error-correction performance was not very obvious, and a code-word with a stronger error-correction ability is required. It can be seen from the above analysis that the RS (15, 9) code can be used in applications where the atmospheric laser-communication system has a small bit-error rate in sunny and cloudy days.

(2) RS (30, 10) code

The error-correction of the system using RS (30, 10) was observed. On sunny and cloudy days, the system error rate after error correction is less than 10^{-9} . The code rate is maintained at the same level 10^{-4} , and the bit error performance after error correction can be achieved 10^{-7} . Generally, the influence of rainy weather on the atmospheric channel can be completely eliminated by RS (30, 10). Under heavy rain, heavy fog, or heavy snow, the bit-error rate of the system is 10^{-2} – 10^{-3} , and the bit-error rate after error correction remains between 10^{-5} – 10^{-6} , which shows that the error-correction ability of the RS (30, 10) code is much better than that of the RS (15,9) code.

(3) LDPC code

The LDPC code used in the measurement had a code length of 128 and a code rate of 1/2. The decoding method was hard-decision bit-flip decoding based on belief propagation.

On sunny and cloudy days, when the LDPC code was used for channel coding, the bit-error rate of the system before error correction was 10^{-7} , and the bit-error rate of the system after error correction was 10^{-9} lower than that, which shows that the system has better error-correction performance. In heavy rain or snowy weather, the system bit-error rate before error correction was maintained between 10^{-2} – 10^{-3} , and the system performance after error correction was 10^{-4} – 10^{-5} . The error-correction improvement was not obvious, and the performance after error correction was not ideal.

The system performance after the error correction is slightly better than that after the RS (15, 9) code was used on the channel, which is basically the same as that of the RS (30, 10) code. Therefore, according to Table 6.2, it can be concluded that LDPC codes with a code length of 128 and code rate of 1/2 are more suitable for occasions where the bit-error rate is not very high. For LDPC codes, obtaining codewords with stronger error-correction abilities requires longer numbers.

(4) Turbo codes

A turbo code with an interleaving length of 64 and 1/3 code rate was used as the atmospheric laser-communication channel code. It can be seen from Table 6.2 that the error-correction performance of the turbo code is superior to that of the RS (15, 9) code and the LDPC code, which are essentially equivalent to the error-correction ability of the RS (30, 10) shortened cyclic code, and can be used in the case of poor wireless laser-communication channel conditions.

When the atmospheric-channel conditions are good, that is, when the system bit-error rate is small, the error-correction performance of the turbo code is much better than that of the RS (15, 9) and LDPC codes. The error rate of the system after the error correction of the RS (15, 9) code and LDPC code is in 10^{-6} , and the performance of the system after the error correction of the turbo code is achieved 10^{-7} . It can be concluded that the turbo code is suitable when the system bit-error rate is 10^{-4} , but for a wireless laser system with a bit-error rate less than 10^{-3} , the error correction ability is generally not as good as RS (30, 10).

6.4 Analysis of the Influence of Meteorological Conditions on the System Bit-Error Rate

When the weather is fine and sunny, the system has the lowest bit-error rate and the best communication quality. Cloudy weather has little impact on the system performance; the atmospheric channel is stable, and the temperature difference in cloudy weather is not large. It is not necessary to consider atmospheric turbulence

on the system bit-error rate in the case of medium and light rain, as the atmospheric channel has little effect on the communication link.

Atmospheric channel-coding technology can be used to improve the performance of the atmospheric channel. In heavy rain, heavy snow, and heavy fog, atmospheric laser communication may be disrupted in bad weather conditions, although rain and snow can hinder the transmission of light; however, the worst environment is fog.

Figure 6.22 shows the change in bit-error rate before and after error correction for several error-correction codes under different weather conditions. The truncated cyclic code RS (30, 10) has the best error-correction performance. The turbo code and its error-correction performance are slightly inferior. Different error-correction methods have different improvement effects on atmospheric channels. In this manner, an appropriate error-control method is selected, according to the measurement results, to improve the characteristics of the atmospheric channel, so that the system has better communication performance under severe weather conditions, such as rain, fog, and snow.

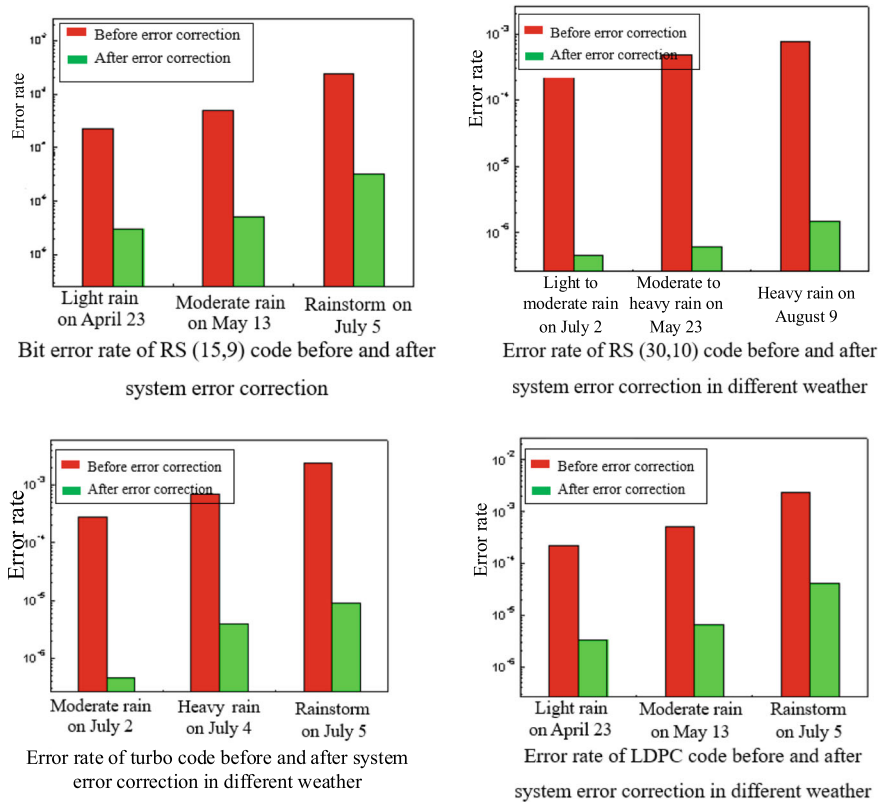


Fig. 6.22 Comparison of the effect of error-correction codes under different weather conditions

From the above analysis and Table 6.2, it can be seen that the error-correction capabilities of the RS (15, 9) code and the LDPC code with a code length of 128 and a code rate of 1/2 are not significantly different. Both can be used in atmospheric laser-communication systems in the occasions where the bit-error rate is small, such as sunny, cloudy, and light rain. They can also be properly applied to moderate to heavy rain; however, the effect is not very obvious, and the system performance after error correction is not very good.

When the interleaving length is 64, the turbo code with a code rate of 1/3 has a strong error-correction ability, and is suitable for atmospheric laser-communication systems with a bit-error rate of less than 10^{-4} . The cyclic-shortened code RS (30, 10) has the same error-correction performance as the turbo code under certain weather conditions, such as sunny, cloudy, and rainy days; however, in severe weather conditions, such as heavy fog, heavy snow, and heavy rain, the turbo code error-correction performance is better.

6.5 Attenuation of a Laser Signal in Rain

We used the test-system setup shown in Figs. 6.7 and 6.8 to test the various conditions. The attenuation of the laser signal after passing through the rain is abnormal. In a light rain or drizzle, the attenuation of the light intensity is greater than that in heavy rain, as shown in Figs. 6.23 and 6.24.

This phenomenon contradicts the formula for rain attenuation that is often used. It is generally believed that the light-intensity attenuation of heavy rain is greater than that of light rain; however, our observational facts do not support this conclusion. From a light-scattering point of view, we attempt to explain the results observed in our experiments.

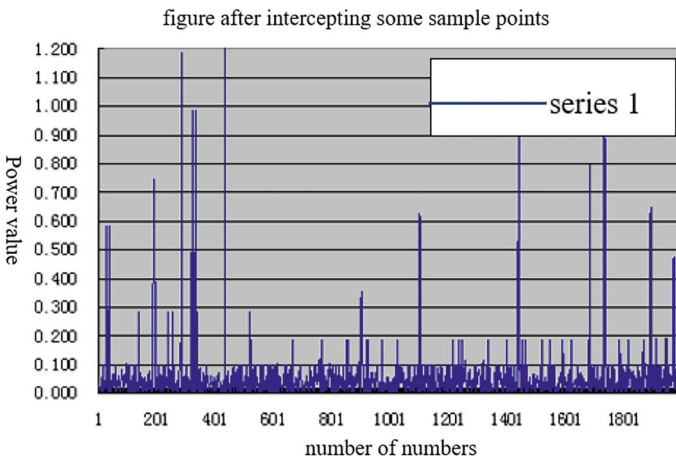


Fig. 6.23 Transmission power curve of a laser in light rain [21]

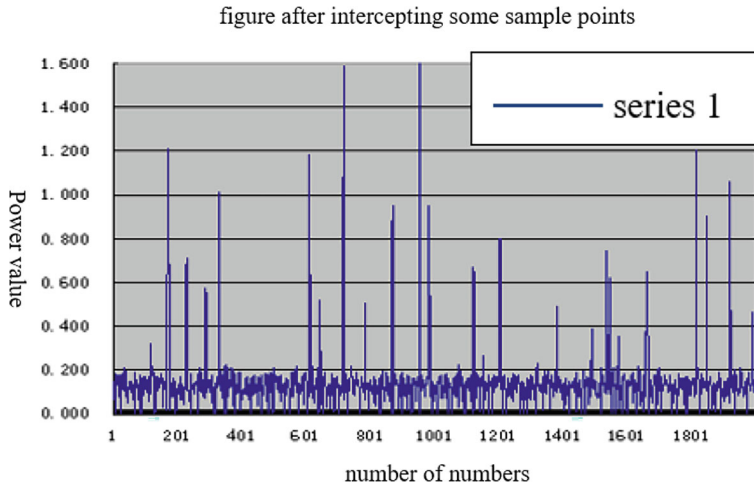


Fig. 6.24 Transmission-power curve of a laser signal in heavy rain [21]

6.5.1 Single-Raindrop Forward Scattering [33–35]

In 1908, Mie applied electromagnetic field theory to describe the scattering of an incident monochromatic plane wave, using a uniform sphere of any size and a composition in a uniform medium, and obtained a strict analytical solution. The Mie scattering solution is not only applicable to the scattering of a single sphere, but also to the scattering problem of multiple spheres. As long as the spheres have the same diameter and composition, and are randomly distributed, they are separated from each other by a distance much larger than the wavelength of the incident light. At this time, the scattered light of different spheres has no definite phase relationship and the total scattered energy is the sum of the scattered energies of each sphere. Therefore, the Mie theory has a wide range of applications when studying the propagation of light in the atmosphere, water, clouds, rain, fog, and aerosols.

According to the Mie theory, when the light intensity I_0 of a laser light with wavelength λ in a medium surrounding the particle is parallel to an isotropic spherical particle with diameter D , the scattered light intensity at scattering angle θ and distance l from the scatter is

$$I = \frac{\lambda^2}{8\pi^2 l^2} I_0 (i_1 + i_2), \quad (6.27)$$

where $i_1 = S_1(m, \theta, \alpha) \cdot S_1^*(m, \theta, \alpha)$ and $i_2 = S_2(m, \theta, \alpha) \cdot S_2^*(m, \theta, \alpha)$. i_1 and i_2 are the intensity functions of scattered light; S_1 and S_2 are the amplitude functions of scattered light. S_1^* and S_2^* are the complex conjugates of S_1 and S_2 , respectively. α ($\alpha = \pi D/\lambda$) is the particle-size parameter and $m = m_1 - im_2$ is the refractive index of the particle relative to the surrounding medium. When the imaginary part is

nonzero, it means that the particle is partially absorbed. S_1 and S_2 can be represented as the following infinite series:

$$S_1 = \sum_{n=1}^{\infty} \frac{2n+1}{n(n+1)} (a_n \pi_n + b_n \tau_n) \quad (6.28)$$

$$S_2 = \sum_{n=1}^{\infty} \frac{2n+1}{n(n+1)} (a_n \tau_n + b_n \pi_n). \quad (6.29)$$

In the formula, a_n and b_n are called the Mie coefficients, which can be calculated using the following formulas:

$$b_n = \frac{m\psi_n(\alpha)\psi'_n(m\alpha) - \psi'_n(\alpha)\psi_n(m\alpha)}{m\zeta_n(\alpha)\psi'_n(m\alpha) - \zeta'_n(\alpha)\psi_n(m\alpha)} \quad (6.30)$$

$$a_n = \frac{\psi_n(\alpha)\psi'_n(m\alpha) - m\psi'_n(\alpha)\psi_n(m\alpha)}{\zeta_n(\alpha)\psi'_n(m\alpha) - m\zeta'_n(\alpha)\psi_n(m\alpha)}, \quad (6.31)$$

where $\pi_n = P_n^{(1)}(\cos(\theta))/\sin(\theta) = dP_n(\cos(\theta))/\sin(\theta)$

$$\tau_n = \frac{d}{d\theta} P_n^{(1)}(\cos(\theta))$$

$$\psi_n(z) = \left(\frac{z\pi}{2}\right)^{1/2} J_{n+(1/2)}(z) \quad (6.32)$$

$$\zeta_n(z) = \left(\frac{z\pi}{2}\right)^{1/2} H_{n+(1/2)}^{(2)}(z), \quad (6.33)$$

where $P_n^{(1)}(\cos(\theta))$ is the first-order n th first-class associative Legendre function and $P_n(\cos(\theta))$ is the first-class Legendre function. $J_{n+(1/2)}(z)$ and $H_{n+(1/2)}^{(2)}(z)$ represent the Bessel function of the half-integer order and the Hankel function of the second kind, respectively.

6.5.2 Single-Sphere Particle Computer Simulation [36–42]

The diameter of a raindrop is between 100 μm and 10 mm, and is generally not more than 8 mm. Raindrops larger than 8 mm are unstable and break. Photographic studies [44] have shown that raindrops with a radius less than 1 mm are spherical. When the light wavelengths are 532-nm visible light and 1064-nm infrared light, their corresponding complex refractive indices of water [44, 45] are $m = 1.332 - i1.96 \times 10^{-9}$ and $1.327 - i2.89 \times 10^{-6}$, respectively. For incident-light power $I_0 = 0.05$ W, the relationship between the logarithmic scattered light intensity and the scattering angle at a distance of $r = 0.005$ m from the scatter.

(1) **Simulation of the relationship between the logarithmic-scattering light intensity and scattering angle of raindrop particles in light waves at different wavelengths**

We performed experimental simulations of incident light waves with wavelengths of 532 and 1064 nm. Assuming that the equivalent radii of raindrop particles are 50 μm and 1 mm, the experimental simulation results of the relationship between the logarithmic scattered light intensity and scattering angle are shown in Figs. 6.25 and 6.26.

It can be seen from the simulation results that

1. When the light-wave wavelengths are the same, the scattered light intensity in all directions of raindrop particles with a large radius is smaller than that of particles with a small radius;
2. When the light-wave wavelength and the particle radius are the same, the forward and backward scattering of the light wave are obviously larger than the side scattering. The scattering energy is mainly concentrated in the forward and backward directions;
3. When the particle radius of the raindrop is the same as the wavelength, the concentration of the forward and backward scattering increases with the increase of the wavelength.

(2) **Simulation of forward-scattered light intensity of different raindrop particle radii for the same light wavelength**

We simulated the forward-scattered light intensity of different raindrop particles with a wavelength of 532 nm, assuming that the scattering angle $\theta = 0^\circ$. The simulation results are shown in Figs. 6.27 and 6.28.

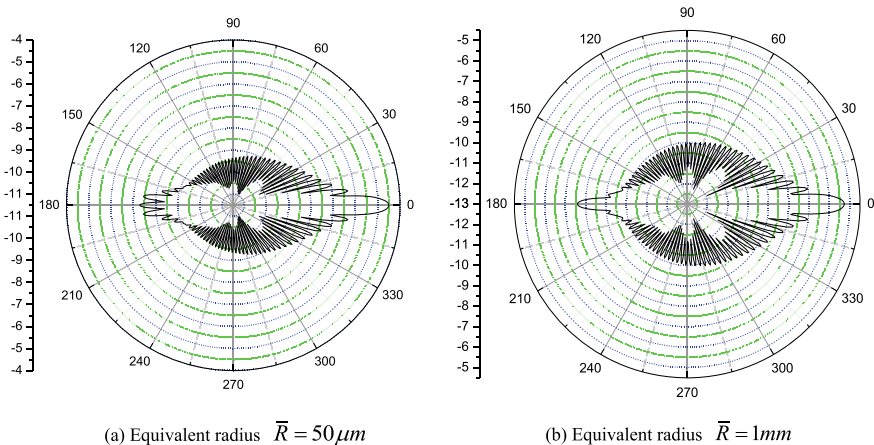


Fig. 6.25 Logarithmic scattered light intensity for incident-light wavelength $\lambda = 532 \text{ nm}$

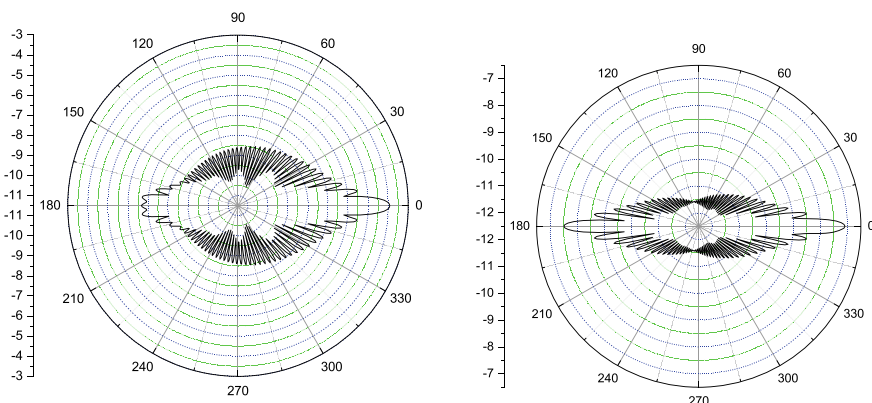
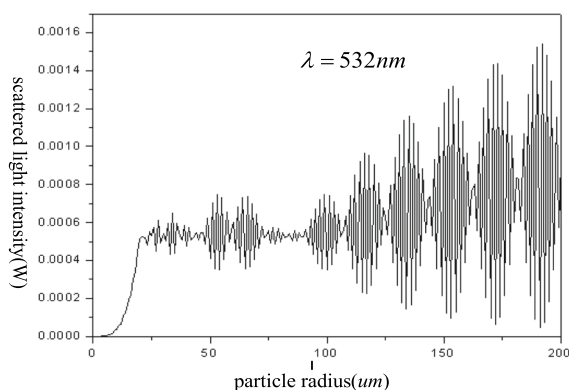
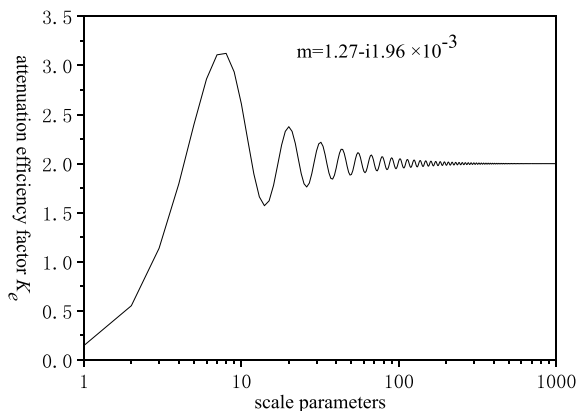
(a) Equivalent radius $\bar{R} = 50\mu\text{m}$ (b) Equivalent radius $\bar{R} = 1\text{mm}$ **Fig. 6.26** Logarithmic scattered light intensity for incident-light wavelength $\lambda = 1064\text{ nm}$ **Fig. 6.27** Forward-scattered light intensity and rain particle radius

Figure 6.27 shows that the forward-scattered light intensity increases with an increase in the particle radius. The overall trend is increasing; however, there is an obvious up-and-down oscillation. The simulation results in Fig. 6.28 show that the attenuation efficiency factor K_e oscillates up and down, with 2 as the center, as the scale parameter changes. The shape of the curve becomes complex, with a series of maximum and minimum values appearing. When the scale parameter is relatively small, K_e increases with the scale.

The attenuation increases as the scale parameter increases, which is equivalent to the Rayleigh approximation. When the scale parameter is less than 105, the attenuation coefficient changes greatly, and when the scale parameter is greater than 105, the change tends to be gentle as the scale parameter increases. This means that the

Fig. 6.28 Attenuation coefficient and scale parameter



attenuation efficiency-factor cannot be taken as 2 in the entire scale range to simplify the calculation, when the Mie single-scattering theory is used to calculate the rain attenuation.

6.5.3 Attenuation of Sparse Rain Particles

(1) Single scattering

Single-particle scattering [43] assumes that there are no other scatterers around the particle. In principle, if two or more scatters are encountered, the interaction between them should be considered, and specific solutions under different boundary conditions should be obtained. This special solution can only be obtained in certain cases. However, if the distance between particles is sufficiently large, the influence of other particles can be completely ignored when considering the scattering of a certain particle. This is called independent scattering or single scattering.

It is generally considered that when the distance between particles is three times the particle diameter, it is sufficient for independent scattering. Considering the concentration of particles in the medium, when the distribution is quite sparse, electromagnetic waves in the atmospheric-transmission path are only scattered by a relatively small number of particles. The scattered field is only due to a single scattering of particles, and all secondary and multiple scatterings can be ignored. As the particle concentration increases, the interaction between particles must be considered, and multiple-scattering approximation needs to be considered.

Figure 6.29 shows a schematic diagram of single scattering. Figure 6.30 shows a schematic diagram of multiple scattering. The particle directly scatters the incident wave to the observation point as the primary scattering. Secondary scattering means that the first particle scatters the incident wave to a second particle, and then the second particle scatters the scattered wave of the first particle to the observation point.

Fig. 6.29 Schematic diagram of single scattering

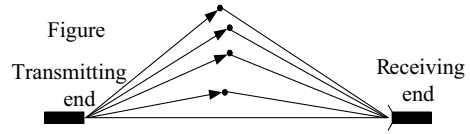
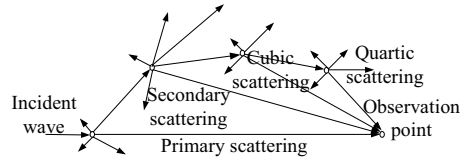


Fig. 6.30 Schematic diagram of multiple scattering



(2) Calculation and simulation of rain attenuation [43]

The raindrop-size distribution is the spatial distribution of raindrops of different sizes, corresponding to different rainfall rates; it is also known as the raindrop spectrum. The earliest raindrop-size distribution measurement was made by Wiesner in 1895 with the “absorbent paper method” [44], and the “flour method” has been widely used since then.

The most famous raindrop-size distribution measurement using the “flour method” was by Laws and Parsons. They carried out extensive measurements of the raindrop-size distribution of different types of rainfall in the Washington area of the United States, and found that, even with the same rainfall rate, the change of the raindrop-size distribution is huge. The distributions were averaged to obtain the mean raindrop-size distribution for different rainfall rates. This is called the Laws–Parsons distribution [45]. The Laws–Parsons raindrop-size distribution is still considered to be the most typical average raindrop-size distribution and is widely used. The International Telecommunication Union Radiocommunication Sector (ITU-R) rain-attenuation model was obtained using the Laws–Parsons distribution.

Joss et al. found that the raindrop-size distribution varies greatly with the type of rainfall. They divided the types of rainfall into drizzle, widespread, and thunderstorm [7]:

$$\text{Drizzle : } n(r) = 60000e^{-5.7R^{-0.21}r} \quad (\text{m}^{-3}\text{mm}^{-1})$$

$$\text{Widespread rain : } n(r) = 14000e^{-4.1R^{-0.21}r} \quad (\text{m}^{-3}\text{mm}^{-1})$$

$$\text{Thunderstorm : } n(r) = 2800e^{-3.0R^{-0.21}r} \quad (\text{m}^{-3}\text{mm}^{-1}),$$

where R is the rainfall rate and r is the rain-particle radius.

The range of rainfall rates corresponding to various rainfall types can be found in the literature [34]. A drizzle is 0.25 mm/h, light rain is 1 mm/h, and moderate rain is 4 mm/h. Heavy rain is 16 mm/h, and a rainstorm is 100 mm/h. Using the raindrop

distribution by Joss et al., computer simulations were carried out for different rainfall rates, and the attenuation coefficient of the rain and the attenuation percentage per unit distance were obtained.

We performed experimental simulations on different types of rainfall, such as drizzle, extensive rain (moderate and heavy rain), and thunderstorms, as shown in Figs. 6.31 and 6.32. It can be seen that the attenuation coefficient of a drizzle or light rain is the largest, and the attenuation coefficient of a thunderstorm (rainfall rate greater than 100 mm/h) is the smallest. This also verifies the experimental phenomenon that the attenuation of light intensity is larger in a light rain or drizzle, than in the case of heavy rain [43, 44].

For large-scale particles, such as raindrops, the scattered light is mainly concentrated in the forward and backward scattering directions, and the scattered light intensity is concentrated in a narrow forward and backward angle. It can be seen from the previous analysis that the raindrop scattering has a significant influence on

Fig. 6.31 Attenuation coefficient of different rainfall rates

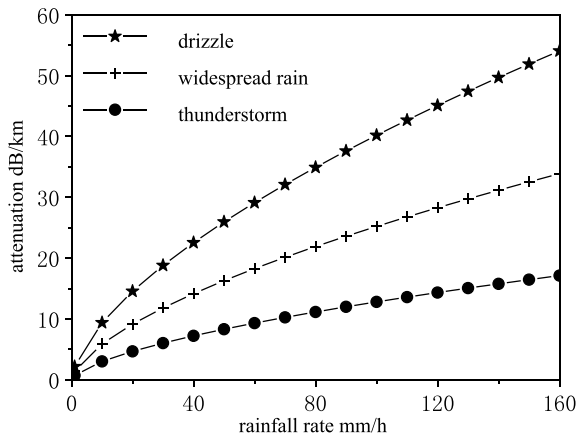
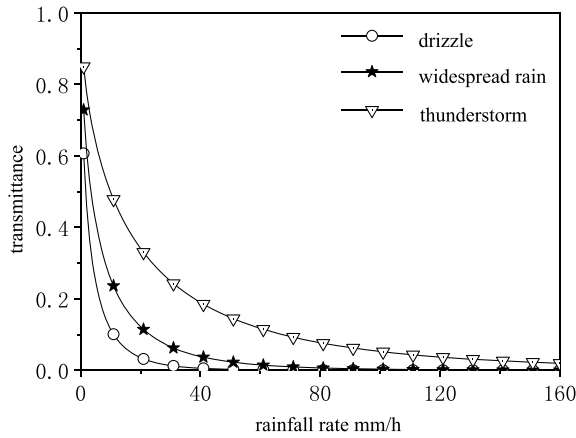


Fig. 6.32 Transmittance of laser light to rain per unit distance (km)



the attenuation of laser light in rain. The results show that the attenuation of a drizzle or light rain has the greatest effect, the attenuation of extensive rain (moderate and heavy rain) is between that of heavy and light rain, and the effect of a thunderstorm on the attenuation is relatively small. This finding was consistent with our observations.

References

1. Ke XZ, Xi XL (2004) The introduction of wireless laser communication, 1st edn. Beijing University of Posts and Telecommunications Press, pp 8–15
2. Liu T, Zhu J, Chen YB et al (2000) The new generation laser communication system in atmosphere. 6(3):131–133
3. Wang LY (2002) The effects of atmospheric conditions on the performance of line-of-sight optical free space communication system. Guangxi University, Guangxi
4. Hu Y, Liu H (1998) Technology and development of free-space laser communication. J Univ Electron Sci Technol 27(5):453–461
5. Li ZT (1997) Discussion of the effect of atmosphere turbulence on propagating laser beam. J Appl Opt 18(3):30–32
6. Wu J, Le SX (1988) Theory of light propagation in random media. Chengdu Institute of Telecommunication Engineering Press
7. Yang CH et al (1990) Laser and infrared technology manual. National Defense Industry Press, Beijing
8. Tatarsky BH, Wen JS, Song FZ et al (1978) Wave propagation theory in turbulent atmosphere. Science Press
9. Lawrence RS, Strohbehn JW (1970) A survey of clear-air propagation effects relevant to optical communications. Proc IEEE 58(10):1523–1545
10. Wisely DR, Mccullagh MJ, Eardley PL et al (1994) 4km Terrestrial line-of-sight optical free-space link operating at 155Mb/s. SPIE 21(23):108–117
11. Strickland BR, Lavan MJ, Woodbridge E et al (1999) Effects of fog on the bit-error rate of a free-space laser communication system. Appl Opt 38(3):424–431
12. Kim II, Woodbridge EL, Chan VJ et al (1998) Scintillation measurements performed during the limited-visibility lasercom experiment. Proc SPIE 32(6):209–220
13. Kim II, Koontz J, Hakakha H et al (1998) Measurement of scintillation and link margin for the TerraLink laser communication system. Proc SPIE 32(32):100–118
14. Kruse P (1962) Elements of infrared technology. Wiley
15. Lutomirski RF, Yura HT (1971) Wave structure function and mutual coherence function of an optical wave in a turbulent atmosphere. J Opt Soc Am 61(4):482–487
16. Rao RZ, Wang SP, Gong Z (1998) Statistical characteristics of laser intensity fluctuation and beam pattern in a real turbulent atmosphere. Chin J Quant Electron 15(2):155–163
17. Chen YB, Zhao SH, Xia GJ (2003) Analysis of dissipation in laser atmospheric transmission. Radio Commun Technol 29(1):1–3
18. Andrews LC, Phillips RL, Hopen CY et al (1999) Theory of optical scintillation. J Opt Soc Am A 16(6):1417–1429
19. Li SG (1999) Effect on laser propagation in the atmospheric turbulence. Changchun Institute of Optics and Mechanics, ChangChun
20. Chen LX (2005) Experimental measurement of atmosphere laser communication system. Xi'an University of Technology, Xi'an
21. Ding DQ (2005) Design of PPM for laser communication in atmosphere. Xi'an University of Technology, Xi'an
22. Du AY (2005) Study and implementation of reed-solomon code in an atmospheric laser communication system. Xi'an University of Technology, Xi'an

23. Wang LL (2006) Numerical calculation and simulation research for atmospheric laser communication. Xi'an University of Technology, Xi'an
24. Gong ZB (1998) Some research progress on high-energy laser propagation in atmosphere. 15(2):114–133
25. Ren H, Wei HJ, Tan JC (2003) Computer simulation of the detecting noise caused by atmospheric scintillation in laser communication. J Appl Opt 24(5):22–24
26. Wang SP, Rao RZ, Liu XC (1998) Experimental observation of laser scintillation effect in real atmosphere. Opto-Electron Eng 25(6):121–124
27. Ke XZ, Yang LH, Ma D (2009) Transmitted attenuation of laser signal in rain. Infrared Laser Eng 37(06):1021–1024
28. Du AY, Ke XZ (2006) Study on atmosphere channel that influencing on laser PPM signal. Laser J 27(1):73–74
29. Zhao L, Ke XZ, Liu J (2007) Study on model and key technique of atmosphere laser communication system. Acta Photonica Sinica S1:27–30
30. Shi SX, Liu JF, Sun YL (2006) Electromagnetic theory of light-propagation and control of light. Xi Dian University Press
31. Huang RH, Zhou SJ et al (trans) (1986) Wave propagation and scattering in random media. Science Press, Beijing
32. Li WM, Ao FL, Yu SY (2006) Study of forward scattering effections on laser transformation in raindrop. J Photon Technol 12(04):237–240
33. Oguchi T (1983) Eletromagnetic wave propagation and scattering in rain and other hydrometers. Proc IEEE 71(9):1029–1078
34. Laws JO, Parsons DA (1943) The relation of raindrop-size to intensity. EOS Trans Am Geophys Union 24(2):248–262
35. Zhao ZW (2001) Study on radio wave propagation characteristics and remote sensing of water condensate. Xi'an University of Electronic Science and technology, Xi'an
36. Liu J, Ke XZ, Zhao L et al (2007) Experimental measurement of atmospheric laser communication. Acta Photonica Sinica S1:10–13
37. Wang LL, Ke XZ, Chen LX (2005) The testing system for optical power through atmosphere. Chin J Light Scatter 17(4):378–383
38. Wang LL, Ke XZ (2005) Real-time display for laser via atmosphere random channel light intensity distribution. J Xi'an Univ Technol 21(2):196–199
39. Wang LL, Ke XZ (2005) Analyze and calculate for laser transmit through rain. Chin J Light Scatter 17(2):148–153
40. Wang R (2007) Study on propagation and attenuation characteristics of laser in fog coal. Xi'an University of Electronic Science and technology, Xi'an
41. Li YJ (2005) Theoretical and experimental study on particle concentration and particle size measurement based on Mie scattering. Zhong Bei University, Shanxi
42. Sun T (2004) Numerical simulation and experimental study on measurement of micro spherical particle size based on Mie scattering theory. Tian Jin University, Tianjin
43. DAd Wolf (2001) "On the Laws-Parsons distribution of raindrop sizes", in Radio Science, July–August 36(4):639–642. <https://doi.org/10.1029/2000RS002369>
44. Levi LW (1980) Applied Optics, 4

Chapter 7

Adaptive Coding Based on Channel Estimation



Light propagation in the atmosphere is subject to attenuation caused by absorption, scattering, and refraction by air; this is especially serious in rainy, snowy, and foggy weather, leading to a sharp increase in the bit-error rate (BER) and even communication interruptions [1–7]. In the case of long communication distances, especially for satellite—ground optical wireless communication, the BER will increase, even if the weather conditions are good, because of the influence of atmospheric scintillation caused by atmospheric turbulence, which is one of the main constraints that limit the communication distance of optical wireless communication [8–13].

At present, there are corresponding solutions to the above problems, such as increasing the optical power and reducing the information rate to combat atmospheric attenuation, using spatial-diversity technology to combat atmospheric scintillation, and using a microwave communication backup, in case of a communication interruption. Although these measures can improve the reliability of optical wireless communication to a certain extent, they are all constrained by geographic conditions, location, environment, cost, human-eye safety, and other factors.

Therefore, to ensure the reliability of space optical-communication systems and further improve the system anti-interference capability, it is necessary to adopt adaptive-modulation coding technology [14, 15], which automatically changes the coding method according to the change in the channel to improve the reliability of the communication system.

7.1 Adaptive-Modulation Coding

7.1.1 Adaptive-Modulation Coding Strategy

Adaptive-modulation coding techniques are used to adjust the modulation coding scheme according to the channel state. Adaptive coding and modulation are based on a relatively accurate estimation of the parameters associated with the channel. When the receiver can estimate the channel-state information and feed it back to the transmitter, the transmitter can improve the spectrum utilization by adjusting the transmission parameters in real time.

Most modulation and coding techniques do not dynamically adjust according to channel fading. Such non-adaptive systems are designed according to the worst-case scenario of the channel state, to ensure the intended performance in all states of the time-varying channel. This system-design principle results in the channel capacity not being fully utilized. In contrast, adaptive transmission uses more-efficient power and rate transmissions under good channel conditions and can fully utilize channel resources.

An ideal adaptive algorithm can adjust various signal-transmission parameters based on the current channel environment. Channel-state information refers to the information that reflects the channel-fading condition, such as the number of retransmissions, BER, channel signal-to-noise ratio (SNR), and decoder decoding conditions. Usually, the channel-state information we refer to is the false frame rate (BER) and channel SNR. The basic principles of the adaptive strategy are as follows:

1. Define a channel-quality indicator or channel-state information that provides some characteristics about the channel.
2. Adjust the signal-transmission parameters according to the channel-state information.

Several methods exist to represent the state information of a channel. Typical methods include the SNR or signal-to-interference-noise ratio (SINR), which can be obtained from the physical layer. At the link layer, the false packet rate obtained from the cyclic redundancy check (CRC) information can also be used, as can the false bit rate.

(1) Channel-status information based on the BER

In some cases, the channel-state information is related to the error rate of the received data corresponding to the various candidate operating modes. The data-reception conditions are updated and saved until all the operating modes have been trained. There is a correspondence between the operating modes and the BER.

Unlike the original approach, which relies on a theoretical BER curve, this adaptive mode of operation provides clear information regarding the observable link quality for each possible candidate mode of operation. However, this method is limited by the number of code elements in the observation window. It relies on the probabilistic characteristics of the false frame rate (or BER), which requires several

thousand packets to be sent for a given operating mode before a reliable estimate can be obtained; this can significantly slow the adaptive process.

In addition, these methods are service-dependent, which makes it difficult to control the response time of the algorithm, and it is impossible to monitor the channel quality when the user is not sending or receiving services. In particular, in the absence of services, the user loses track of the channel quality and must start a new adaptive process. This method has the advantage of being simple to implement; however, its performance is not ideal because it does not accurately reflect the changes in a time-varying fading channel in a timely manner.

(2) Channel-state information based on the mean signal-to-noise ratio

To achieve adaptive transmission, valid channel-state information must be available at either the transmitter or the receiver side. This information often consists of the SNR measured at the receiver side. In this case, the following adaptive strategies can be adopted:

1. The SNR is measured at the receiver side.
2. Based on each candidate operating mode, the SNR information is converted into the corresponding BER information.
3. Based on the target BER, an operating mode is selected that yields the maximum throughput within the bounds of the guaranteed target BER.
4. The selected operating mode is fed back to the transmitter side.

Step 1 corresponds to the evaluation of the channel-state information. Step 2 involves the calculation of the threshold value of the adaptive algorithm. In this case, the threshold is defined as the minimum SNR required to operate at the target BER for a given operating mode. Step 3 refers to the selection of the optimal operating mode, based on the threshold and measured SNR. Step 4 is concerned with feeding the information back to the transmitter. This adaptive communication-system model is illustrated in Fig. 7.1.

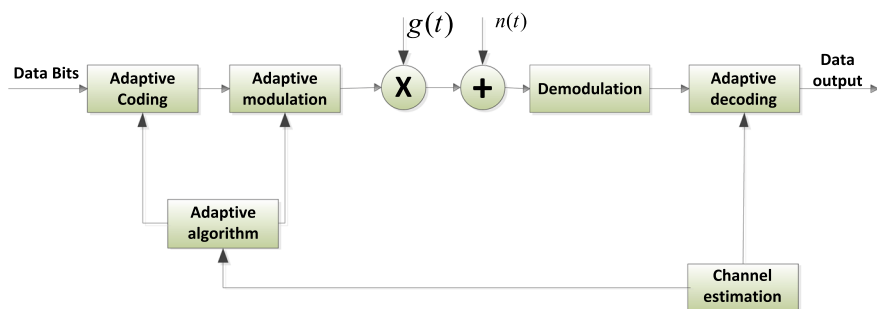


Fig. 7.1 Adaptive communication-system model

7.1.2 *Implementation of Adaptive-Modulation Coding*

The process of implementing adaptive techniques in general communication systems can be divided into the following four steps.

(1) Estimate the channel state [16–19]

The quality of the time-varying channel is estimated to obtain the channel-state information (CSI), and the adaptive technique changes the transmission parameters of the next symbol frame (or time slot), according to the variation in the channel. In practical systems, the channel-transfer function and SNR can be used as short-term channel-state information, and the false bit rate and false frame rate as long-term channel-state information.

Channel-state information can be obtained in different systems using various methods. If the communication between two terminals is bidirectional and the channel can be considered reciprocal, each terminal can estimate the channel-state information, based on the received symbols, and adaptively adjust the local transmission parameters according to the estimation results. This is called the open-loop adaptive method.

If the channel is not reciprocal (as in frequency-division duplex (FDD) systems), the receiver estimates the channel quality and informs the sender of this information over the reverse link; the sender then determines the transmission parameters based on the feedback parameters. This is called closed-loop control. If the communication is not bidirectional, a low-speed signaling channel must be established between the receiver and sender.

(2) Select the threshold value for the coded modulation method [20, 21]

The transmitter coding and modulation method is determined by comparing the channel-quality prediction with the threshold values of various coding and modulation methods. The selection of the threshold value determines the system BER and throughput performance. Therefore, the system performance should be analyzed and studied, according to the system requirements, and the threshold value of the adaptive coding modulation scheme should be reasonably determined according to certain criteria, that is, the transmission SNR interval corresponding to each coding-modulation method.

(3) Control the data transmission at the transmitter

After obtaining the CSI, the transmission characteristics of the transmitter can be changed accordingly; this is the core of adaptive technology. The coding modulation used for the next transmission is determined, based on a comparison of the channel-estimation output with the threshold value, and the transmitter is controlled to encode and modulate the transmitted data using that transmission mode.

(4) **Notify the receiver of the coding-modulation method**

The receiver must know which coding-modulation method is used to demodulate and decode the received signal. Therefore, it is important to correctly identify the coding-modulation method; this is generally accomplished in three ways.

a. **Open-loop method**

The receiver estimates the channel based on the reception and notifies the sender via signaling. Alternatively, using reciprocity, the sender directly estimates the channel, selects parameters based on the channel condition, and notifies the receiver via signaling.

b. **Closed-loop method**

The receiver estimates the channel based on the reception, selects the parameters, and then notifies the sender via signaling.

c. **Blind detection**

When no signal is transmitted, the sender selects the parameters according to its reception, and the receiver detects the transmitted parameters blindly.

From the above analysis, it can be seen that reasonably determining the judgment threshold for various transmission modes in the adaptive-coding modulation system is one of the keys to improving the system performance using the implementation scheme. Various guidelines exist for determining the threshold value from optimizing the system performance. Here, we briefly describe two commonly used ones: One is to ensure that the system has a certain BER performance, and the other is to ensure that the system-throughput performance (or spectrum utilization) is maximized.

We know that different coding-modulation modes in wireless-communication systems have different BER performances. For a transmission mode, when the channel state deteriorates (SNR decreases), the transmission BER will increase, which may hinder the system from transmitting data properly. Therefore, the judgment threshold of each coding-modulation mode can be determined by ensuring that the system has a certain BER. When the channel SNR is lower than the threshold value, its corresponding coding-modulation mode will not be used for system transmission.

7.1.3 Analysis of the Adaptive-Modulation Coding Algorithm

The core issue of adaptive coding is the transmission-mode selection algorithm. To date, the transmission-mode selection algorithm has not been clearly specified in most wireless-communication specifications (e.g., high-speed downlink packet access (HSDPA)), but is given full play to various equipment manufacturers. The R&D departments of various companies and other research institutions have conducted

extensive and in-depth research in this area, and the common transmission-mode selection algorithms can be divided into exploratory and channel-estimation-based adaptive-selection algorithms [14, 15].

(1) Exploratory adaptive

The main idea of the exploratory-adaptive coding algorithm is to track the data frames that fail to transmit, which is generally indicated by the CRC check result indication (ACK/NAK frames), and to transform its data-transmission rate using the tracking result and the system-target indication.

The basic principle is that the number of consecutive failed data frames n and cumulative number of successful data frames m are set according to the system requirements. If n consecutive data frames are not successfully received by the originator, the second retransmission of the current packet and subsequent data transmission are performed with a lower coded-transmission mode.

When the accumulated number of successfully received data frames reaches m , the data-transmission mode is stepped up one level and the accumulator is zeroed. However, if the transmission mode is stepped down one level and the immediately following frame fails again, the transmission mode is stepped down again and the accumulator is zeroed.

The advantage of this algorithm is that it is simpler to implement because it does not require channel estimation, and there is no performance loss owing to channel-estimation errors. However, because of its exploratory nature, it is not able to react quickly to channel changes and the conversion-coding method tends to lag behind the channel changes; thus, it is difficult to greatly improve the system throughput using the exploratory class of algorithms. In addition, the number of consecutive failed data frames n and the number of cumulative successful data frames m in the implementation process must be determined. The system model is shown in Fig. 7.2.

(2) Channel-estimation-based adaptive

Although the exploratory algorithm is simpler to implement, it does not fully adapt to channel changes because the channel information cannot be fully applied without channel estimation; therefore, the transmission-mode selection algorithm based on channel estimation is more studied in adaptive coding. While receiving the data, the receiver estimates and predicts the channel conditions for the next data frame, based on the variation in the received signal, and feeds it back to the transmitter, which then decides the coding mode to be used for the next frame, based on the feedback value, to better adapt to the channel variation.

Adaptive-coding techniques based on channel estimation usually consist of three components: measuring, estimating, or predicting the channel variation; setting the judgment threshold; and signaling the information to the rest of the system. The system model is shown in Fig. 7.3.

The channel-quality estimation requires a suitable physical-quantity representation, and the metrics generally used are the signal-to-noise ratio (SNR) and mean squared error (MSE). These values can be measured at the receiver side (closed-loop

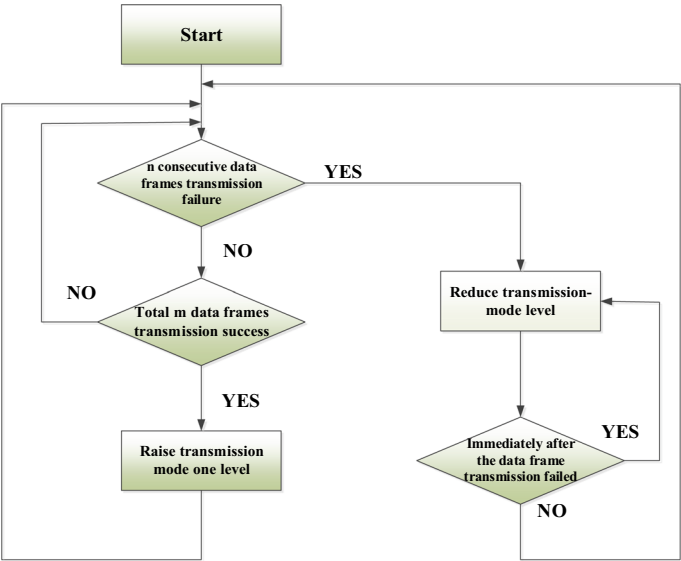


Fig. 7.2 Flowchart of the exploratory-adaptive algorithm

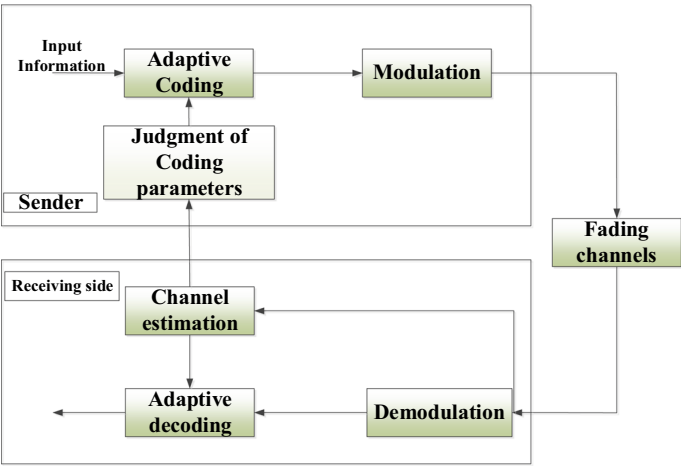


Fig. 7.3 Adaptive-coding system model based on channel estimation

approach) or at the transmitter side (open-loop approach), in addition to considering the adaptive rate and complexity of the estimation algorithm.

If the communication is bidirectional and the channels are shared, the transmitter can estimate the transmission channel based on the received information, thus enabling the corresponding coding mode in the next transmission frame. This

approach is called open-loop detection and is used, for example, for wireless channels in near-ground laser-communication modes.

If the transmission channel is not the same for both sides of the communication, as when it is divided into uplink and downlink, the quality of the channel must be aggregated at the receiver side, and the detection result is communicated to the sender side by means of signaling. This method of transmitting the channel condition using the receiver is called closed-loop detection.

After obtaining the channel-quality information through channel estimation, it is important to consider how to change the coding parameters, such as code rate and code length; that is, to set a suitable selection threshold. Selecting the optimal threshold involves optimizing the objective function under limited conditions. There are two common optimization objectives:

1. Under certain spectrum-utilization conditions, the transmission BER is minimized.
2. To ensure a certain BER, the transmission rate is maximized; that is, spectrum utilization is maximized.

The judgment threshold can also be fixed or varied dynamically. A fixed threshold can be obtained from mathematical calculations or through simulation, which is simpler to implement but usually not optimal. The adaptive-decision threshold can adapt to the time-varying characteristics of the channel, improve the system throughput to a greater extent, and can reach or approach the optimal threshold; however, it increases the system complexity.

After estimating the channel quality and determining the judgment threshold, it is important to inform the sender of this information to ensure that the transmission mode at the sender and receiver is the same. One method is to transmit the channel quality directly in the signaling; another is to signal the sender directly of the transmission mode that it should use, after the receiver has estimated the transmission mode at the sender, based on the channel quality. The latter method reduces the feedback information and is more efficient.

7.2 Adaptive-Channel-Estimation Algorithm

An adaptive-coding strategy applied to optical wireless communication can adjust and select coding/decoding schemes with different error-correction capabilities or change the parameters of the same coding/decoding scheme, in real time, according to the channel conditions during the entire communication process.

High error-correction-strength coding can waste bandwidth and other resources in a clear atmosphere. Low error-correction-strength coding cannot guarantee the communication quality in a bad atmosphere under a single coding method. An adaptive strategy is an effective means of improving the performance of optical wireless communication from the coding perspective by balancing reliability and economy [11, 12].

Because laser communication is bidirectional, each link can estimate the channel using its reverse link. Using the atmospheric laser-communication transceiver's characteristics to integrate full-duplex communication, a training sequence is inserted at the two terminals of the communication link, when sending. The training sequence is first detected at both receiving ends, and then the channel parameters are estimated by the corresponding channel-estimation module.

At the initial moment, the corresponding subencoder is used, according to the initially estimated channel parameters. When the channel characteristics change, the estimated channel parameters will change accordingly. Then, the system automatically changes the coding and decoding of both sides simultaneously, and selects the subencoder that matches the new channel parameters, thereby achieving adaptive coding.

7.2.1 Training-Sequence Design

The training sequences used for channel estimation must have good autocorrelation and correlation properties and be easy to detect. To reduce the system complexity, frame-synchronization codes can be used as training sequences. Pseudo-random sequences are typically used as training sequences for frame-synchronous codes. m -sequences, Gold sequences, and Barker codes are ideal code words. In addition, m -sequences, Gold sequences, and Walsh sequences have been successfully used in wireless radio-frequency (RF) communications and have obtained relatively good correlation and better system performance.

Because the user information is processed in the optical domain and the optical signal is unipolar, m -sequences, Gold sequences, and Walsh sequences cannot be directly applied to the atmospheric laser-communication system and must first be converted to unipolar codes.

(1) m -sequences

The m -sequence can be generated using a linear-feedback shift register. In the binary shift-register generator, if n is the number of steps, the maximum length of the code sequence that can be generated is $2^n - 1$. The m -sequence, as a pseudo-random sequence, has three important properties:

1. In each cycle, code element 1 appears once more than code element 0, which is balanced.
2. In each cycle, there are $2^n - 1$ element trips, half of which are code element 1 and half of which are code element 0.
3. The m -sequence $\{a_j\}$ and its displacement sequence of mode 2 $\{a_{j-t}\}$, and another displacement sequence in the m -sequence.

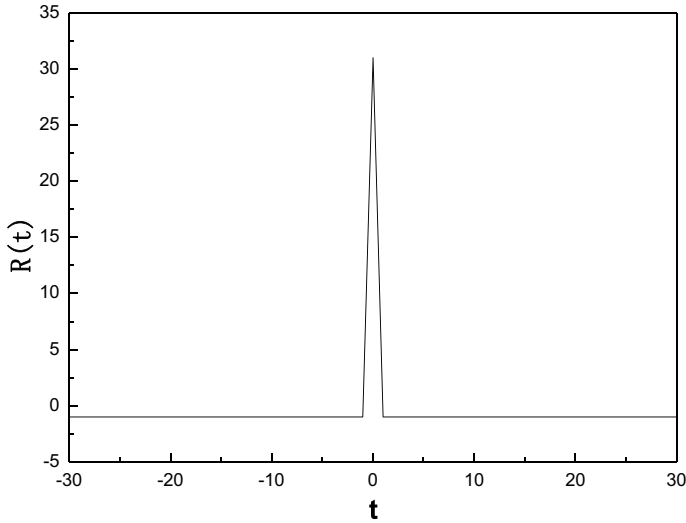


Fig. 7.4 *m*-Sequence autocorrelation properties

These three characteristics of the *m*-sequence give it an excellent autocorrelation function, as shown in Fig. 7.4, with low mutual correlation of the preferred sequence-association set. It has a narrow sense of the pseudo-noise sequence, is easy to generate and replicate, and is the most widely used sequence.

(2) Gold sequences

In 1967, R. Gold proposed a composite sequence composed of preferred pairs, referred to as Gold codes. Gold codes are generated based on *m*-sequences and consist of a modulo-2 sum of two *m*-sequence preferred pairs with equal code lengths and the same code clock [22].

Let $a_i(t) \in (-1, 1)$ denote a Gold code with length T_g , number of code bits N , and width of code slice T_e ; that is, $T_g = NT_e$ and $a_i(t) = 0(t > T_g, t < 0)$. The Gold code is defined as

$$R_{ijk}(\tau) = \int_0^T [a_i(t + \tau) + a_j(t - T_g + \tau)] a_k(t) dt \quad (0 \leq \tau \leq T_g). \quad (7.1)$$

1. When $i = j = k$, $R_{iii}(\tau)$ is said to be the periodic autocorrelation function of $a_i(t)$.
2. When $i = j \neq k$, $R_{iik}(\tau)$ is said to be the periodic intercorrelation function of $a_i(t)$ and $a_k(t)$.

For a binary Gold-code sequence, the correlation function has the following formula:

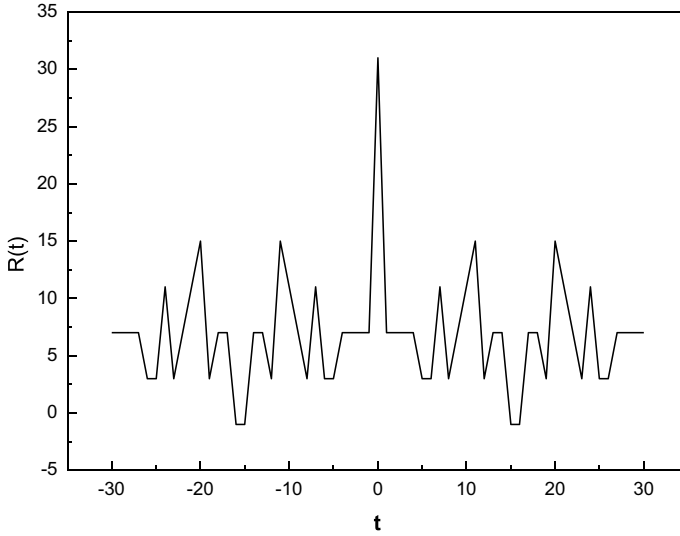


Fig. 7.5 Autocorrelation property of a Gold sequence

$$R_{ijk}(r) = \sum_{m=0}^{N-1} [a_i(m+r) + a_j(m-N+r)]a_k(m) \quad (0 \leq r \leq N-1), \quad (7.2)$$

where $a_j(m) = a_j(t = mT_e)$, $(0 \leq m \leq (N-1))$.

Setting $N = 15$, the following sequence of Gold codes is obtained by calculation:

$$x = \{111101011001000\}. \quad (7.3)$$

When Gold codes are correlated with a long random sequence containing Gold codes, the correlation value is only larger when they are aligned with each other, as shown in Figs. 7.5 and 7.6. When Gold codes are correlated with a long random sequence containing Gold codes, the correlation value is larger only when the Gold codes are encountered in the long sequence.

7.2.2 Channel Estimation

Channel estimation is the process of estimating the parameters of a channel model from the received data. Channel estimation plays a key role in wireless systems and is typically required to provide channel-state information to the receiver.

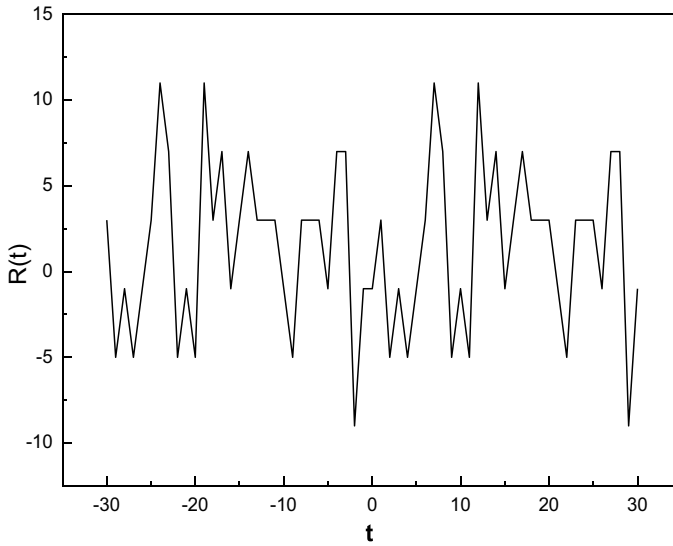


Fig. 7.6 Gold-sequence correlation properties

(1) Estimation-theory basis

The estimation theory generally addresses the stochastic phenomenon, based on an analysis of the observed data after contamination by various noises and disturbances, and is further based on certain estimation criteria to estimate the random variables or stochastic processes. If the quantity to be estimated is a random variable, the estimation method is called parametric estimation; if the quantity to be estimated is a random process, it is called state or waveform estimation. The estimation theory consists of two parts: parametric estimation and waveform estimation [23].

Among the estimation theories, Bayesian estimation requires a known cost function, along with a knowledge of the complete probabilistic description of the parametric quantities to be estimated, and observed data under the most demanding conditions. The maximum a posteriori probability (MAP) and maximum-likelihood (ML) estimations have somewhat relaxed conditions, requiring that the cost function be an even function of the error without its detailed form; however, they still require a knowledge of the complete probabilistic description of the covariates and observations to be estimated. Linear least mean-squared-error (LMMSE) estimation requires only the first- and second-order moment properties of the parameter to be estimated and the observed data to be known. Least-squares (LS) estimation treats the estimation problem as an optimization problem.

(2) Classification of channel-estimation techniques

Channel estimation is a key technique in wireless communication. It was shown in [24] that at a medium SNR (20 dB), a 1% channel-estimation error can degrade the channel capacity by approximately 1/3. High-quality channel estimation is a strong

guarantee of the performance of a communication system. Typically, channel estimation can obtain the frequency-domain transfer function of a wireless channel to help the receiver demodulate the currently received signal. For adaptive-modulation systems, channel estimation can also be used to obtain the SNR of the currently received signal, and thus, predict the possible channel conditions for the next transmission and select the optimal transmission parameters.

Channel-estimation techniques can be broadly classified into two categories: one is pilot-frequency-based channel estimation, which includes both pilot-frequency-based and training-sequence-based methods. The other category is not based on a guide frequency and includes blind estimation and differential estimation.

Blind channel-estimation methods do not need to transmit additional redundant information for channel-estimator training; however, they require constant or slowly changing channel-statistical properties (much smaller than the convergence time of channel estimation) because of the large delay introduced by using a large amount of transmitted data for accurate channel estimation. Otherwise, the channel cannot be estimated accurately.

Semi-blind channel-estimation methods also do not use special training sequences or a small amount of training data to complete the channel estimation, so their channel-estimation convergence time is also very long. These two methods are suitable for channels with statistically smooth or quasi-smooth characteristics and slow time-varying channels.

Pilot-symbol-assisted modulation (PSAM) is a method that uses a pilot frequency (training sequence) to estimate the channel. In general, PSAM channel estimation first inserts a certain amount of known data (pilot frequency) into the data stream, and then estimates the channel response at the time or frequency of the pilot frequency using some criterion. It then obtains the complete channel response by one-dimensional or two-dimensional interpolation. This is the most studied channel-estimation method.

(3) Adaptive channel-estimation algorithms

a. Maximum-likelihood estimation

The estimation model containing a vector of parameters to be estimated is assumed to be as follows [23, 24]:

$$Y = Xh + N, \quad (7.4)$$

where $h = [h_1, \dots, h_M]^T$ is the vector of parameters to be estimated; that is, the channel characteristics at the guide-frequency location. X is a known $M \times N$ matrix, N is an n -dimensional noise vector, and all its elements obey the same independent Gaussian distribution with a zero mean and variance σ_n^2 . $Y = [Y_1, \dots, Y_M]^T$ is the vector of observations; that is, the received signal vector.

To perform a maximum-likelihood (ML) estimation of parameter vector h , the likelihood equation for the maximum-likelihood estimation can be obtained as

$$\frac{\partial}{\partial h} \ln p(y|X, h) \Big|_{h=\hat{h}_{ML}} = 0. \quad (7.5)$$

The maximum-likelihood estimate in Eq. (7.5) is then used to construct a cost function $p(y|X, h)$, such that the cost function achieves the largest value of h as the final maximum-likelihood estimate:

$$\hat{h} = \arg \max_h \{p(y|X, h)\}. \quad (7.6)$$

The final estimate is the \hat{H} that yields the maximum value of this cost function. For the ML method, the discussion is based on the likelihood function of the following equation:

$$f(Y|H) = \frac{1}{(2\pi)^{N/2} |C|^{1/2}} \exp \left\{ -\frac{1}{2} (Y - HX)^H C^{-1} (Y - HX) \right\}, \quad (7.7)$$

where C is the covariance array of noise Z and superscript H denotes the conjugate transpose. The cost function in Eq. (7.7) is used to estimate the corresponding \hat{H} by taking the partial derivative of the cost estimator H and setting it equal to zero. Because the noise is Gaussian white noise, the maximum-likelihood estimate of H can be obtained by simplification:

$$\hat{H}_{ML} = YX^H (X^H X)^{-1}. \quad (7.8)$$

b. Least-squares estimation

Least-squares (LS) channel estimation is a channel estimator obtained from the least-squares sense (LSS) [23, 25, 26]. The estimation model containing the vector of parameters to be estimated is given by the following equation:

$$Y = Xh + N, \quad (7.9)$$

where $h = [h_1, \dots, h_M]^T$ is the vector of parameters to be estimated; that is, the channel characteristics at the guide-frequency location. X is a known $M \times N$ matrix, N is an n -dimensional noise vector, and all its elements obey the same independent Gaussian distribution with a zero mean and variance σ^2 . $Y = [Y_1, \dots, Y_M]^T$ is the vector of observations and also the received signal vector.

From a knowledge of least-squares theory, to complete the least-squares estimation of parameter h , the cost function of the least-squares estimation is first examined:

$$C(h) = (Y - Xh)^H (Y - Xh), \quad (7.10)$$

The cost function in Eq. (7.10) that minimizes \hat{h} is the least-squares estimate of h . Furthermore, by biasing the cost function in Eq. (7.10) with respect to h and setting it equal to 0, the channel response H_{LS} can be obtained by the least-squares criterion:

$$H_{LS} = \min\{(Y - Xh)^H(Y - Xh)\} \quad (7.11)$$

$$\frac{\partial\{(Y - Xh)^H(Y - Xh)\}}{\partial H_{LS}} = 0. \quad (7.12)$$

The time-domain estimate can be obtained as

$$\hat{h}_{LS} = (x^T x)^{-1} x^T y, \quad (7.13)$$

where x is the sending vector and y is the receiving vector.

For frequency-domain estimates,

$$\hat{H}_{LS} = X^{-1} Y \quad (7.14)$$

$$\hat{H}_{LS} = [H(0)H(1) \cdots H(N-1)]^T = X^{-1} Y \quad (7.15)$$

$$= \left[\frac{Y(0)}{X(0)} \frac{Y(1)}{X(1)} \cdots \frac{Y(N-1)}{X(N-1)} \right]^T. \quad (7.16)$$

Here,

$$X = \begin{bmatrix} X(0) & 0 & & \\ & \ddots & & \\ 0 & & & X(N-1) \end{bmatrix} \quad (7.17)$$

is the diagonal matrix of the transmitting sequence vector. The elements on the diagonal are the message symbols transmitted on the corresponding subchannels.

However, the inverse matrix of X does not exist when it is not a square matrix, or when X is a square matrix, but $\det(X) = 0$. To solve the problem that the above matrix cannot be directly inverted, the LS algorithm replaces the direct inverse of matrix X with generalized inverse matrix $(X^H X)^{-1} X^H$, such that the expression of the channel-estimation result is

$$\hat{H}_{LS} = (X^H X)^{-1} X^H Y. \quad (7.18)$$

The LS criterion-based channel-estimation algorithm has a simple structure, performs only one division operation, and requires little computational effort. However, the correlation properties of the channel frequency and time domains are not utilized in LS estimation, and the channel estimates are sensitive to the effects of noise because the effects of noise are ignored. With a small SNR, the estimation accuracy is low, which affects the parameter estimation of the data subchannel.

From the above analysis, it can be seen that in the case where the noise is additive Gaussian white noise (AGWN), maximum-likelihood estimation and least-squares estimation are equivalent and have the same expression form. Of course, if the noise is not AGWN, the expression of the maximum-likelihood estimation cannot be reduced to the form of least squares, but is much more complicated.

Under the assumption of Gaussian white noise, the first- and second-order statistical properties can fully describe the stochastic process, and both maximum likelihood and least squares can obtain all the statistical information of the quantity to be estimated, based on the first- and second-order statistical properties; thus, they can obtain the same estimation results.

c. Minimum mean-squared-error estimation

The minimum mean-squared-error (MMSE) algorithm is based on the Bayesian method for statistically significant estimation [26–28], which models the channel impact response h as a random vector. In the derivation according to the MMSE criterion, we must assume that h is a Gaussian variable with zero mean and is uncorrelated with the noise variable N . Under these assumptions, we can obtain

$$\hat{g}_{MMSE} = R_{gy} R_{yy}^{-1} y. \quad (7.19)$$

Here,

$$\begin{aligned} R_{gy} &= E\{gy^H\} = R_{gg} F^H X^H \\ R_{yy} &= E\{yy^H\} = XFR_{gg}F^H X^H + \sigma_n^2 I_N \end{aligned} \quad (7.20)$$

are the mutual-covariance matrix of g and y and the self-covariance matrix of y , respectively; R_{gg} is the self-covariance matrix of g and σ_n^2 denotes the noise variance $E\{|n_k|^2\}$. These two statistics are assumed to be known. Because the columns in F are orthogonal, the frequency-domain MMSE estimate \hat{h}_{MMSE} can be derived from \hat{g}_{MMSE} :

$$\hat{h}_{MMSE} = F \hat{g}_{MMSE} = F Q_{MMSE} F^H X^H y. \quad (7.21)$$

Here, Q_{MMSE} can be represented as

$$Q_{MMSE} = R_{gg} [(F^H X^H F)^{-1} \sigma_n^2 + R_{gg}]^{-1} (F^H X^H X F)^{-1}. \quad (7.22)$$

If g is not Gaussian, \hat{h}_{MMSE} does not need to be the minimum mean-squared-error estimate; it is still the best linear estimate, and in any case (whether g is Gaussian or not), we use \hat{h}_{MMSE} to denote the channel estimate.

MMSE estimation has good performance but relatively high complexity, whereas LS estimation has low complexity, but does not perform as well as MMSE. It should

be noted that MMSE estimation is predicated on the assumption that the channel self-covariance and noise variance are known. In practice, these statistics, R_{gg} and σ_n^2 , also need to be corrected or estimated, possibly adaptively. This increases the estimation complexity and slightly degrades the performance.

d. Channel estimation using Wiener filtering based on an MMSE criterion

If signal $x(n)$ or $X(n)$ and observed data $z(n)$ or $Z(n)$ are generalized smooth and their autocorrelation functions or power-population densities are known or findable, the linear minimum mean-squared-error estimation criterion is used to estimate $x(n)$ or $X(n)$. This estimation is called Wiener filtering. The basic idea of Wiener filtering is to find the optimal shock function or transfer function of a linear filter that minimizes the mean-squared error between the output and input waveforms [23, 24, 27], as shown in Fig. 7.7.

The estimated value of the hypothetical channel is [29]

$$\hat{H} = \sum_{i=1}^n w_{k,j} Y_i = W_k^H Y = Y^H W_k. \quad (7.23)$$

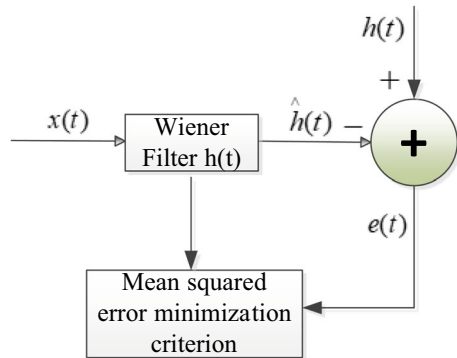
The estimation error is

$$\begin{aligned} e_k &= H_k - \hat{H}_k \\ &= H_k - W_k^H Y = H_k - Y^H W_k \end{aligned} \quad (7.24)$$

$$\begin{aligned} E[e_k \cdot e_k^H] &= E\left[(H_k - \hat{H}_k) \cdot (H_k - \hat{H}_k)^H\right] \\ &= E[H_k H_k^H] - 2E[H_k Y^H] W_k + W_k^H E[Y Y^H] W_k \end{aligned} \quad (7.25)$$

Let $R_{yy} = E[Y Y^H]$ and $R_{hy} = E[H_k Y^H]$; then,

Fig. 7.7 Block diagram of the Wiener filtering system



$$E[|e_k|^2] = E[|H_k|^2] - 2R_{hy}W_k + W_k^H R_{yy} W_k. \quad (7.26)$$

In Eq. (7.26), to take the minimum value of $E[|e_k|^2]$, we find the partial derivative of W_k , satisfying

$$\begin{aligned} \frac{\partial E[|e_k|^2]}{\partial W_k} &= 2R_{yy}W_k - 2R_{hy} = 0 \\ \Rightarrow W_k &= R_{yy}^{-1}R_{hy}. \end{aligned} \quad (7.27)$$

Therefore,

$$\hat{H} = W_k^H Y = R_{hyk}^H R_{YY}^{-1} Y \quad (7.28)$$

$$\hat{H} = E[HH^H]R_{yy}^{-1}Y = R_{HY}R_{YY}^{-1}Y, \quad (7.29)$$

in which,

$$R_{HY} = E[HY^H] = E[H(XH + N)^H] = R_{HH}X^H \quad (7.30)$$

$$\begin{aligned} R_{YY} &= E[YY^H] = E[(XH + N)(XH + N)^H] \\ &= XR_{HH}X^H + \sigma_N^2 I_N. \end{aligned} \quad (7.31)$$

Therefore,

$$\begin{aligned} \hat{H} &= R_{HY}R_{YY}^{-1}Y = R_{HH}X^H(XR_{HH}X^H + \sigma^2 I_N)^{-1}Y \\ &= R_{HH}[X^{-1}(XR_{HH}X^H + \sigma^2 I_N)(X^H)^{-1}]^{-1}X^{-1}Y \\ &= R_{HH}(R_{HH} + \sigma^2(X^H X)^{-1})^{-1}X^{-1}Y \\ &= R_{HH}(R_{HH} + \sigma^2(X^H X)^{-1})^{-1}\hat{H}_{LS} \end{aligned} \quad (7.32)$$

From the above discussion, the channel-estimation expression based on the MMSE criterion for Wiener filtering can be obtained as

$$\hat{H} = R_{HH}(R_{HH} + \sigma^2(X^H X)^{-1})^{-1}\hat{H}_{LS}, \quad (7.33)$$

where R_{HH} is the autocorrelation matrix corresponding to the channel impact and σ^2 is the variance of the additive Gaussian noise. Whenever signal X changes, matrix $R_{HH} = E[HH^H]$ must change accordingly. To further reduce the complexity, $(X^H X)^{-1}$ can be replaced by its expected value, $E\{(X^H X)^{-1}\}$.

When the signal is modulated with equal probability,

$$E\{(X^H X)^{-1}\} = E\{|1/x_k|^2\}I, \quad (7.34)$$

where I is the unit matrix.

We define the average SNR as $E\{|x_k|^2\}/\sigma^2$, and further simplify it to obtain

$$\hat{H} = R_{HH} \left(R_{HH} + \frac{\beta}{SNR} I \right)^{-1} \hat{H}_{LS}. \quad (7.35)$$

Here, $SNR = E|X_p(k)|^2/\sigma_n^2$ is the average signal-to-noise power ratio and $\beta = E|X_k(k)|^2 E|1/X_p(k)|^2$ is a constant that depends on the modulation method. If the self-covariance matrices R_{HH} and SNR are known a priori, $R_{HH} \left(R_{HH} + \frac{\beta}{SNR} I \right)^{-1}$ need only be calculated once.

e. Estimation based on singular-value decomposition

Although the linear minimum mean-squared-error estimator uses only the frequency-domain correlation property and its complexity is lower than that of estimators that use both time- and frequency-domain correlations, it still requires a large number of complex computational procedures. The channel matrix may be singular in the process of finding the inverse, which makes it impossible to find the inverse, or the wrong value, and is therefore limited in practical applications.

In contrast, the singular-value decomposition (SVD) algorithm can be obtained by a low-order approximation of the matrix transformation and can be obtained, regardless of whether the channel matrix is singular or not, thus also reducing the complexity of the implementation [28, 30, 31].

A singular-value decomposition of the frequency-domain channel vector self-covariance matrix R_{HH} yields

$$R_{HH} = U \Lambda U^H, \quad (7.36)$$

where U is a matrix consisting of normalized orthogonal vectors and Λ is a diagonal matrix containing the singular values $\lambda_0 \geq \lambda_1 \geq \dots \geq \lambda_{N-1} \geq 0$ of R_{HH} . Substituting the above equation into $\hat{H} = R_{HH} \left(R_{HH} + \frac{\beta}{SNR} I \right)^{-1} \hat{H}_{LS}$, we obtain the channel estimator based on singular-value decomposition, as follows:

$$\hat{H}_{SVD} = U \Delta U^H \hat{H}_{LS}, \quad (7.37)$$

where Δ is a diagonal matrix whose values on the diagonal are

$$\delta_k = \frac{\lambda_k}{\lambda_k + \frac{\beta}{SNR}} \quad k = 0, 1, \dots, N-1. \quad (7.38)$$

By replacing part of matrix Δ on the diagonal with 0, a simplified approximate estimate can be obtained as

$$\hat{H}_{SVD} = U \begin{bmatrix} \Delta_J & 0 \\ 0 & 0 \end{bmatrix} U^H \hat{H}_{LS}, \quad (7.39)$$

where Δ_J is the upper-left corner of the $J \times J$ matrix in Δ . Changing the size of J allows some compromise between complexity and performance.

f. Interpolation-based channel estimation

When the lead-in interval is smaller than the coherent bandwidth of the channel, the frequency-domain response of the entire channel transmitting data information can be estimated by interpolation in the frequency domain. Thus, the frequency domain of the received signal can be equalized and the information data can be recovered. Different interpolation algorithms obtain different estimation accuracies for the channel frequency-domain response.

The interpolation method is used for fitting channel information, and its most important feature is its simplicity of implementation. This can be considered an optimization of the estimated channel response. Interpolation includes several methods, such as linear interpolation, Gaussian interpolation, etc. The performance is sequentially higher and the complexity increases. However, because the training phase contains noise, interpolation introduces new noise and produces a noise-threshold (error-floor) effect.

The noise brought about by the interpolation method depends on two aspects: first, the number and method of lead insertion. The greater the number of leads, the better the data obtained by interpolation, and the better the corresponding noise-cancellation effect; however, this also reduces the effectiveness of the transmitted data. The second aspect depends on the use of interpolation methods, such as the previously mentioned linear interpolation and Gaussian interpolation methods. The interpolation method can be improved in the same manner as the guide frequency to obtain better results.

At this point, two popular one-dimensional (1D) interpolation filters are discussed and analyzed [27].

g. Linear interpolation

Linear interpolation, which uses two leading-frequency signals for interpolation estimation, is one of the simplest and most commonly used methods:

$$\hat{H}(k) = \hat{H}(mL + l) = \left[1 - \frac{l}{L} \right] \hat{H}_p(m) + \frac{l}{L} \hat{H}_p(m + 1). \quad (7.40)$$

At lower SNRs, linear interpolation is simple to implement and can achieve good performance over LS estimation. This is because lower SNRs imply high noise intensity and larger errors in LS estimation, whereas linear interpolation uses interpolation methods for channel estimation, which can reduce some noise effects.

h. Lagrangian interpolation

Lagrangian interpolation is a curve-fitting method based on Lagrangian polynomials. It is a relatively flexible method based on multiple polynomials, and the order of the polynomials can be chosen arbitrarily. Polynomials of corresponding complexity can be selected to complete the interpolation, according to different performance requirements; thus, it is a very practical method that can use limited resources to the maximum extent.

Given a set of data $(X_0, Y_0) \cdots \cdots (X_{N_L-1}, Y_{N_L-1})$, the formula for Lagrangian interpolation is

$$f(x) = \sum_{i=0}^{N_L-1} f(x_i) L_i(x), \quad (7.41)$$

where N_L is the order of the Lagrangian polynomial, that is, the order of the Lagrangian interpolation; $L_i(x)$ is the Lagrangian polynomial:

$$L_i(x) = \frac{\prod_{k=0, k \neq i}^{N_L-1} (x - x_k)}{\prod_{k=0, k \neq i}^{N_L-1} (x_i - x_k)}. \quad (7.42)$$

From the above analysis, the following can be obtained concluded:

1. In terms of complexity, linear interpolation utilizes two lead-frequency (training-sequence) signals and has lower computational complexity, whereas Lagrangian interpolation utilizes multiple lead-frequency signals, thus resulting in higher computational complexity.
2. Lagrangian interpolation has better performance because it can better approximate the channel-response curve, owing to the large number of leads utilized.

7.2.3 Performance Simulation and Result Analysis

We used on-off keying (OOK) modulation with an atmospheric-scintillation factor of 0.6, assuming a perfectly aligned optical path, and various algorithms to estimate the simulation results, as shown in Fig. 7.8 [32].

From the simulation results, the best performance was obtained using SVD estimation, followed by Weiner-filter estimation, LS estimation, and MMSE estimation. The mean squared error can converge under several estimation methods. However, simulations indicate that there is still a relatively large discrepancy in the performance between each estimation result and the estimation results obtained in wireless communication. After analyzing the simulation data, this appears to be due to the unipolarity of the training sequence, which causes a certain energy loss in the estimation; thus, resulting in a larger estimation error in optical wireless communication systems.

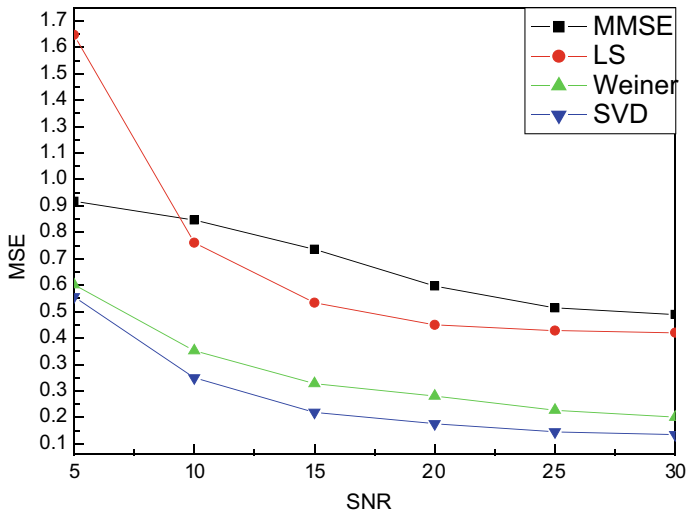


Fig. 7.8 Performance of various estimation algorithms

7.2.4 Improved Channel-Estimation Algorithm

Because the optical signal is unipolar and can only take values in the $(0, 1)$ domain, the sent training sequences are unipolar sequences consisting of 0 and 1. Therefore, the various estimation algorithms described earlier were performed using OOK and pulse-position modulation (PPM) with unipolar training sequences for channel characterization.

As a result, because of the specificity of the training sequence, any channel response multiplied by a sequence containing “0” is still “0” at the corresponding “0” element. This results in a large number of “0” points in the estimated channel characteristics, regardless of the channel variation; this causes a relatively large energy loss and a relatively large estimation error.

The following is an analysis of the LS algorithm as an example, from Eq. (7.4):

$$Y = Xh + N.$$

It can be observed that signal X is transmitted through channel $X \cdot h$ and reaches the receiver after additional Gaussian white noise interference, such as background noise. From the training sequence $x = \{111101011001000\}$, after passing through the atmospheric channel to the receiver side, the value of Y at the corresponding “0” point of x will only have the value of noise N . Furthermore, from Eq. (7.18),

$$\hat{H}_{LS} = (X^H X)^{-1} X^H Y.$$

The LS estimation results are also known to be affected.

From the above analysis, it is concluded that several estimation algorithms that are popular in wireless communication cannot be directly applied in optical wireless communication. Therefore, some improvements were made to compensate for the estimation errors, so that they can be better applied in optical wireless communication systems.

During the signal propagation, the channel changes very little within one signal-propagation period and can be approximated as having no change. This means that the channel characteristics have a strong correlation within one signal period. Accordingly, the channel vector \hat{H}_{LS} , estimated using Eq. (7.18), can be vector-decomposed to extract the non-“0” elements to obtain

$$\left\{ \hat{H}_{LS}(1), \hat{H}_{LS}(2), \hat{H}_{LS}(3), \hat{H}_{LS}(4), \hat{H}_{LS}(6), \hat{H}_{LS}(8), \hat{H}_{LS}(9), \hat{H}_{LS}(12) \right\}.$$

Finding its mean value $E(\cdot)$ yields

$$H_{LS_Mean} = E \left\{ \hat{H}_{LS}(1), \hat{H}_{LS}(2), \hat{H}_{LS}(3), \hat{H}_{LS}(4), \hat{H}_{LS}(6), \hat{H}_{LS}(8), \hat{H}_{LS}(9), \hat{H}_{LS}(12) \right\}. \quad (7.43)$$

Substituting H_{LS_Mean} for the corresponding “0” point of the estimated value of \hat{H}_{LS} , we obtain

$$\hat{H}_{LS} = \left\{ \begin{array}{l} \hat{H}_{LS}(1), \hat{H}_{LS}(2), \hat{H}_{LS}(3), \hat{H}_{LS}(4), H_{LS_Mean}, \hat{H}_{LS}(6), H_{LS_Mean}, \hat{H}_{LS}(8), \hat{H}_{LS}(9), \\ H_{LS_Mean}, H_{LS_Mean}, \hat{H}_{LS}(12), H_{LS_Mean}, H_{LS_Mean}, H_{LS_Mean} \end{array} \right\}. \quad (7.44)$$

A performance comparison before and after the improvement is shown in Fig. 7.9a, b.

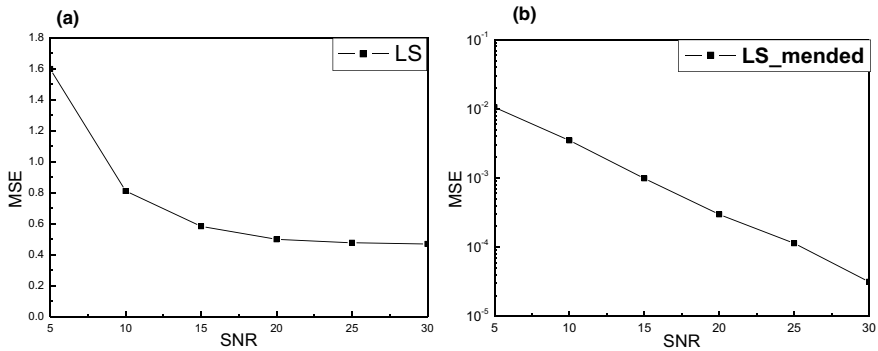


Fig. 7.9 LS algorithm performance-comparison graph. **a** Unimproved; **b** improved [32]

Fig. 7.10 Performance of each algorithm after improvement [32]

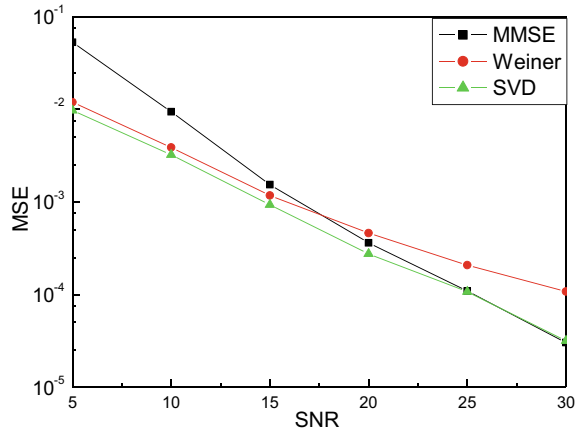


Figure 7.9a, b show the results obtained using LS estimation before and after correction. From the simulation results, the MSE of the LS estimation with correction can be reached at 20 dB, and the performance is improved by three orders of magnitude. The same improvement can be applied to other algorithms, and a performance comparison is shown in Fig. 7.10.

From the simulation results, the MSEs estimated by the various estimation algorithms after the correction were also greatly improved, and the convergence degrees were also relatively improved. The minimum mean squared error (MSE) can be improved by approximately one to two orders of magnitude at small SNRs and by approximately three orders of magnitude at large SNRs.

7.3 Signal-to-Noise Ratio Estimation Algorithm

The signal-to-noise ratio (SNR) can be used to measure the channel quality, describe the wireless-channel quality, provide communication-system channel-quality information, and help improve the communication-system performance. In an adaptive-coding communication system, an appropriate coding method can be selected, based on the SNR obtained from real-time estimation.

The S/N estimation algorithm is based on different channels and signal-modulation methods. There are two main categories: S/N estimation methods based on training-aid data and blind S/N estimation algorithms.

7.3.1 SNR Estimation Based on Subspace Autocorrelation

After the photoelectric conversion at the receiving end, the signal is amplified and filtered before entering the judgment circuit, which can be expressed as

$$r(n) = s(n)h(n) + w(n), \quad (7.45)$$

where $r(n)$ before entering the judgment circuit, can be seen as a continuous signal with varying light-intensity amplitude, owing to noise, as observed by an oscilloscope. Therefore, analytical calculations were conducted [33].

Because the received signal $r(n)$ contains $s(n)$, $h(n)$, and $w(n)$ components that are independent of each other, the autocorrelation matrix of the received signal can be expressed as

$$\begin{aligned} R_{rr} &= E\{r(n)r(n)^H\} = E\{[s(n)h(n) + w(n)] \cdot [s(n)h(n) + w(n)]^H\} \\ &= E\{[s(n)h(n)] \cdot [s(n)h(n)]^H\} + E\{w(n)w(n)^H\} \\ &= R_{ss}R_{hh} + R_{ww} \end{aligned} \quad (7.46)$$

Let $R_{xx} = R_{ss}R_{hh}$; then, Eq. (7.46) can again be expressed as $R_{rr} = R_{xx} + R_{ww}$. Because the autocorrelation matrices R_{rr} , R_{xx} , and R_{ww} are all symmetric matrices, the singular-value decomposition can be performed as follows:

$$R_{xx} = v\Lambda_x v^H, \quad (7.47)$$

where v is the orthogonal matrix and $\Lambda_x = \text{diag}(r_1, r_2, \dots, r_p, 0, 0, 0)_{m \times m}$, where $\text{diag}(\cdot)$ denotes a matrix with \cdot as the diagonal element, $r_1 \geq r_2 \geq \dots \geq r_p$, and p is less than the matrix order m .

$$R_{ww} = v\Lambda_w v^H, \quad (7.48)$$

where $\Lambda_w = \text{diag}(\sigma_w^2, \sigma_w^2, \dots, \sigma_w^2)_{m \times m}$.

$$R_{rr} = v\Lambda_r v^H = R_{xx} + R_{ww} = v(\Lambda_x + \Lambda_w)v^H. \quad (7.49)$$

where $\Lambda_r = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_m)_{m \times m}$

$$= \begin{bmatrix} r_1 + \sigma_w^2 & 0 & \dots & 0 \\ 0 & \ddots & & \\ \vdots & & r_1 + \sigma_w^2 & \ddots \\ 0 & \dots & 0 & \sigma_w^2 \end{bmatrix}_{m \times m} \quad (7.50)$$

We call the space tensed by the above m singular values the noise-containing signal space, the space tensed by the first p singular values the received signal subspace, and the space tensed by the last $m-p$ singular values the noise subspace. It can be shown that the $x(n)$ power P_s can be expressed as the sum of the singular values, and the noise $w(n)$ power P_w is m times σ_w^2 [34]. Thus, the received SNR can be calculated as

$$SNR = 10 \log(P_s/P_w) = 10 \log \left(\frac{\sum_{k=1}^p (\lambda_k - \sigma_w^2)}{m \times \sigma_w^2} \right) \quad (7.51)$$

The sequence of correlation functions $\{\hat{r}_0, \hat{r}_1, \dots, \hat{r}_{m-1}\}$ is estimated from the received signal $y(n)$, and the estimation equation can be expressed as follows:

$$\hat{r}_k = E[y(n)y(n+k)] = \frac{1}{N} \sum_{i=0}^{N-1} y(i)y(i+k) \quad k = 0, 1, 2, \dots, m-1. \quad (7.52)$$

where N denotes the length of the received sequence. As the optical signal is a unipolar real signal, $\hat{r}_k = \hat{r}_k^*$, where $*$ denotes the conjugate transpose. Therefore,

$$\hat{R}_{yy} = \begin{bmatrix} \hat{r}_0 & \hat{r}_1 & \cdots & \hat{r}_{m-1} \\ \hat{r}_1 & \hat{r}_0 & \cdots & \hat{r}_{m-2} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{r}_{m-1} & \hat{r}_{m-2} & \cdots & \hat{r}_0 \end{bmatrix}. \quad (7.53)$$

Calculating the singular-value decomposition of \hat{R}_{yy} yields a sequence of singular values,

$$|\hat{\lambda}_1 \geq \hat{\lambda}_2 \geq \dots \geq \hat{\lambda}_m|. \quad (7.54)$$

The average power of each noise harmonic is calculated using the following equation:

$$\hat{\sigma}_w^2 = \frac{1}{\hat{m} - \hat{p}} \sum_{i=\hat{p}+1}^{\hat{m}} \hat{\lambda}_i. \quad (7.55)$$

The SNR of the received signal was obtained by substituting Eq. (7.51).

7.3.2 SNR Estimation Based on Noise-Variance Estimation

Usually, channel-state information (CSI) is defined as the SNR at the carrier position, and the common method for estimating the SNR is to estimate the signal power and noise power separately, and then divide them by each other. In the case of AGWN channels, the CSI is calculated, and it is important to perform both signal and noise-power estimation to improve the performance. Here, we estimated the signal and noise powers in an atmospheric channel [35, 36].

The value of the SNR can be calculated by Eq. (7.56):

$$CSI = \left| \hat{h}_k \right|^2 / \sigma_n^2 \quad (7.56)$$

$$SNR_k = \frac{\left\langle \left| \hat{H}(k) \right|^2 \cdot |X(k)|^2 \right\rangle}{\sigma_k^2}. \quad (7.57)$$

where the noise power σ_k^2 value is obtained using Eq. (7.58).

$$\left\langle \left| \hat{H}(k) - \hat{H}'(k) \right|^2 \right\rangle = \left\langle \left| H(k) + \frac{W(k)}{X(k)} - \left[H(k) + \frac{W'(k)}{X(k)} \right] \right|^2 \right\rangle, \quad (7.58)$$

where \hat{H} denotes the channel estimate of the current symbol and \hat{H}' denotes the channel estimate of the previous symbol at frequency k .

$W(k)$ and $W'(k)$ are the Fourier transforms of the current and previous AGWN, respectively, which are independently and identically distributed; thus, Eq. (7.58) can become

$$\left\langle \left| \hat{H}(k) - \hat{H}'(k) \right|^2 \right\rangle = 2 \cdot \frac{\langle |W(k)|^2 \rangle}{\langle |X(k)|^2 \rangle}. \quad (7.59)$$

For the noise power,

$$\sigma_k^2 = \langle |W(k)|^2 \rangle = 2 \cdot \langle |X(k)|^2 \rangle \cdot \left\langle \left| \hat{H}(k) - \hat{H}'(k) \right|^2 \right\rangle. \quad (7.60)$$

Substituting σ_k^2 into Eq. (7.57), we obtain

$$SNR = \frac{\left\langle \left| \hat{H}(k) \right|^2 \right\rangle}{2 \cdot \left\langle \left| \hat{H}(k) - \hat{H}'(k) \right|^2 \right\rangle}. \quad (7.61)$$

Therefore, the SNR can be calculated using the channel estimate.

7.3.3 SNR Estimation Based on the Least-Squares Criterion

By the least-squares criterion, the signal and noise can be defined as $\hat{H}X$ and ε , respectively, corresponding to the \hat{H} that minimizes the mean squared error shown in Eq. (7.62) [37]:

$$\varepsilon^2 = \left\langle \left| Y - \hat{H}X \right|^2 \right\rangle, \quad (7.62)$$

where $\langle \cdot \rangle$ represents the ensemble average and $\hat{H}X$ is used as the reference signal for the observed signal at the receiver. Expanding (7.62), we observe that minimizing Eq. (7.62) is equivalent to minimizing Eq. (7.63).

$$|H|^2 \langle |X|^2 \rangle - HR_{XY} - HR_{YX}, \quad (7.63)$$

where R_{XY} denotes the correlation operation, and the correlation function is defined as

$$R_{XY} = R_{YX} = \langle XY \rangle. \quad (7.64)$$

Set

$$T = |H|^2 \langle |X|^2 \rangle - HR_{XY} - HR_{YX}. \quad (7.65)$$

By taking the partial derivative of both sides of Eq. (7.65) and setting it equal to zero, we obtain

$$\frac{\partial T}{\partial H} = \frac{\partial (|H|^2 \langle |X|^2 \rangle - HR_{XY} - HR_{YX})}{\partial H} = 0. \quad (7.66)$$

The unique solution of channel coefficient H can be obtained as

$$\hat{H} = \frac{R_{XY}}{\langle |X|^2 \rangle}. \quad (7.67)$$

Therefore, the SNR estimate of the ratio of the signal power to the noise power, expressed by Eq. (7.62), can be expressed as

$$\hat{SNR} = \frac{|\hat{H}|^2 \langle X \rangle^2}{\varepsilon^2}. \quad (7.68)$$

After simple processing and substitution, Eq. (7.68) can be expressed in the following form:

$$\hat{SNR} = \frac{|R_{XY}|^2}{\langle |Y|^2 \rangle \langle |X|^2 \rangle - |R_{XY}|^2} = \frac{|R_{XY}|^2}{R_{YY}R_{XX} - R_{XY}^2}. \quad (7.69)$$

The least-squares criterion-based SNR estimation is then obtained from Eq. (7.69).

7.3.4 SNR Estimation Using a Correlation-Coefficient Solution

Two sequences, $X(n)$ and $Y(n)$, acquired under the same conditions, contain the same real signal and the same variance of the noise signal. Therefore, they can be set as

$$X(n) = S(n) + W_1(n) \quad (7.70)$$

$$Y(n) = S(n) + W_2(n), \quad (7.71)$$

where $X(n)$ and $Y(n)$ are received sequence 1 and received sequence 2, respectively, and S is the signal; W_1 and W_2 are the noise signals with variance σ^2 . The values of correlation functions $r_{xy}(m)$ of $X(n)$ and $Y(n)$ at $m = 0$,

$$\begin{aligned} r_{xy}(m)|_{m=0} &= \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=0}^{N-1} [X(n)Y(n)] \\ &= \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=0}^{N-1} \{[S(n) + W_1(n)][S(n) + W_2(n)]\} \\ &= r_{ss}(0) + r_{w_1s}(0) + r_{sw_2}(0) + r_{w_1w_2}(0) \\ &= r_{ss}(0) + 2r_{sw_1}(0) + 0 \end{aligned}$$

$$\approx r_{ss}(0) = r_s(0) = E_s, \quad (7.72)$$

where $r_{w1s} = r_{sw2} = 0$ because $S(n)$, $W_1(n)$, and $W_2(n)$ are uncorrelated. $r_s(0)$ is the value of the autocorrelation function $r_s(m)$ of $S(n)$ at $m = 0$, which is equal to the average power E_s of $S(n)$; therefore,

$$r_{xy}(0) = r_s(0) = E_s. \quad (7.73)$$

In addition, because

$$\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=0}^{N-1} X^2(n) = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=0}^{N-1} Y^2(n), \quad (7.74)$$

when N is large,

$$\frac{1}{N} \sum_{n=0}^{N-1} X^2(n) = E_x \cong \frac{1}{N} \sum_{n=0}^{N-1} Y^2(n) = E_y, \quad (7.75)$$

where E_s , E_x , and E_y are the average powers of $S(n)$, $X(n)$, and $Y(n)$, respectively. The correlation coefficients of $X(n)$ and $Y(n)$ are

$$\begin{aligned} \rho_{xy} &= \frac{\frac{1}{N} \sum_{n=0}^{N-1} [X(n) \cdot Y(n)]}{\sqrt{\frac{1}{N} \sum_{n=0}^{N-1} X^2(n)} \cdot \sqrt{\frac{1}{N} \sum_{n=0}^{N-1} Y^2(n)}} \\ &= \frac{r_{xy}(0)}{E_s} \approx \frac{r_s(0)}{E_x} = \frac{r_s(0)}{E_y}; \end{aligned} \quad (7.76)$$

that is,

$$E_s = r_s(0) = \rho_{xy} E_x. \quad (7.77)$$

In addition, because

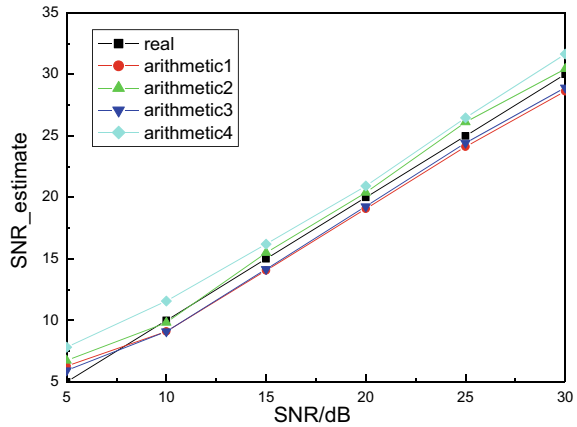
$$E_x = E_s + E_{w1}, \quad (7.78)$$

$$\frac{E_s}{E_{w1}} = \frac{E_s}{E_x - E_s} = \frac{\rho_{xy} E_x}{E_x - \rho_{xy} E_x} = \frac{\rho_{xy}}{1 - \rho_{xy}}. \quad (7.79)$$

Therefore, the SNRs of SNR_x and SNR_y of x and y are

$$SNR_x = SNR_y = 10 \lg \left[\frac{E_s}{E_w} \right] = 10 \lg \left[\frac{\rho_{xy}}{1 - \rho_{xy}} \right]. \quad (7.80)$$

Fig. 7.11 Comparison of four SNR-estimation methods [32]



From the above derivation and analysis, it is clear that if a signal is implied in two sequences with the same variance noise, the SNR of this signal can be estimated by calculating the correlation coefficient of these two sequences [16]. When designing an adaptive-coding system, the SNR of a signal can be estimated and analyzed according to Eqs. (7.76) and (7.80) by obtaining two consecutive received training sequences of signals that contain the same signal and noise variance.

7.3.5 Performance Simulation and Discussion

From the simulation results, the received SNRs estimated by all four algorithms show a relatively good linear relationship, which is in good agreement with the ideal value; thus, they can be used as a basis for the parameters to measure the channel quality in adaptive coding (Fig. 7.11). We also performed simulations under binary phase-shift keying (BPSK) modulation and, because the transmitted signal can be represented as a bipolar signal after BPSK modulation, the simulation results are also given here for added comparison, as shown in Figs. 7.12 and 7.13 [32].

The results obtained from the improved channel estimation are basically consistent in performance with the use of bipolar codes. The improved algorithms provide a good basis for estimating the channel quality in practical applications, with a better solution in terms of a complexity and performance compromise.

7.3.6 Adaptive Judgment Based on Channel Estimation

At the receiving end of the optical wireless communication, after a photoelectric conversion, the signal is amplified and filtered, and then enters the judgment circuit.

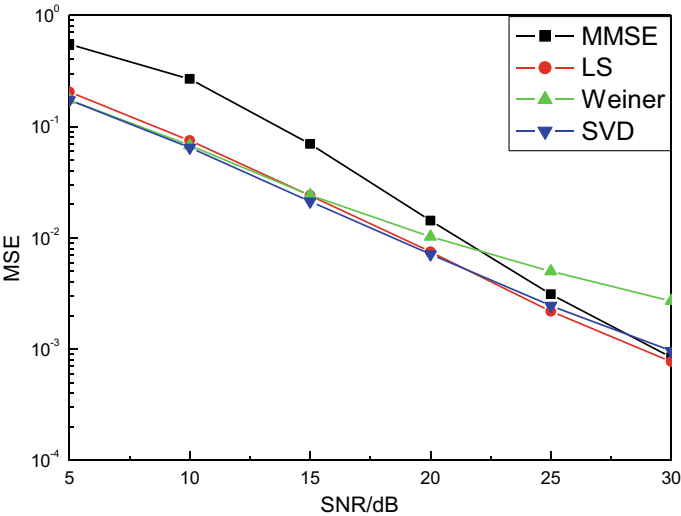


Fig. 7.12 Channel-estimation performance with BPSK modulation [32]

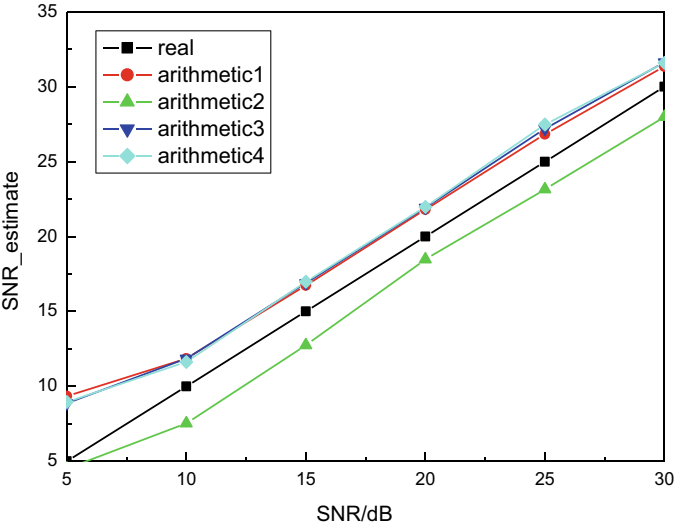


Fig. 7.13 SNR estimation with BPSK modulation [32]

The judgment circuit decides whether the signal is “1” or “0”, according to the judgment threshold. Therefore, the choice of the judgment threshold directly affects the BER of the system.

According to the literature [38, 39], it is known that under wireless-optical OOK modulation, the probabilities of the receiver misclassifying a “1” as “0” and vice

versa are shown in Eqs. (7.81) and (7.82), respectively.

$$P_{0/1} = \frac{1}{2} \left\{ 1 + \operatorname{erf} \left[\left(b - \sqrt{S_t} \right) / \sqrt{2\sigma_n^2} \right] \right\} \quad (7.81)$$

$$P_{1/0} = \frac{1}{2} \left\{ 1 - \operatorname{erf} \left[b / \sqrt{2\sigma_n^2} \right] \right\}, \quad (7.82)$$

where

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-u^2) du = 1 - \operatorname{erfc}(x). \quad (7.83)$$

Then, the total BER is $P_{se} = P_1 P_{0/1} + P_0 P_{1/0}$.

$$P_{se,OOK} = \frac{1}{4} \left\{ \left[1 + \operatorname{erf} \left[\frac{b - E_p}{\sqrt{2}\sigma_n} \right] \right] + \left[1 - \operatorname{erf} \left[\frac{b}{\sqrt{2}\sigma_n} \right] \right] \right\}, \quad (7.84)$$

where b is the judgment threshold, E_p is the transmission power, and σ_n is the noise standard deviation. From Eq. (7.84), it can be seen that the value of the judgment threshold has a significant influence on the BER of the system for a certain transmission power.

Conventional receive verdicts under optical-wireless OOK use fixed verdict thresholds because the atmospheric-channel characteristics are unknown when the verdict is made at the receiver side. Therefore, the OOK verdict in the presence of additive noise is discussed first. Because

$$y(n) = s(n) + w(n), \quad (7.85)$$

the transmission success-rate forms are

$$P_y = \sqrt{S} + P_w, \quad (7.86)$$

where P_y is the received optical power, S is the transmitted optical power, and P_w is the noise power.

Assume that the modulated sources have equal probability of sending “1” and “0”, and $P_0 + P_1 = 1$.

In optical-wireless OOK modulation, the optimal judgment threshold in the absence of turbulence is

$$b = \sqrt{S}/2. \quad (7.87)$$

The judgment threshold shown in Eq. (7.87) is only related to the transmit power, and it is a fixed threshold, which is typically used in conventional OOK-modulation judgments.

However, because the actual atmospheric environment is subject to attenuation caused by turbulence, absorption, etc., the system model shown in Eq. (7.85) should be expressed as

$$y(n) = s(n)g(n) + w(n). \quad (7.88)$$

Similarly, transmission success-rate forms are

$$P_y = \sqrt{gS} + P_w, \quad (7.89)$$

where P_y is the received optical power, S is the transmitted optical power, P_w is the noise power, and g is the attenuation coefficient caused by turbulence, absorption, etc. Therefore, the fixed threshold value calculated using Eq. (7.87) adds a new error to the BER calculation using Eq. (7.84).

By introducing the adaptive-judgment threshold, Eq. (7.87) can be expressed as

$$b = \sqrt{gS}/2, \quad (7.90)$$

where the value of g can be obtained from the channel estimation.

We used a Monte-Carlo simulation to compare the system performance with fixed and adaptive judgment thresholds, as shown in Fig. 7.14.

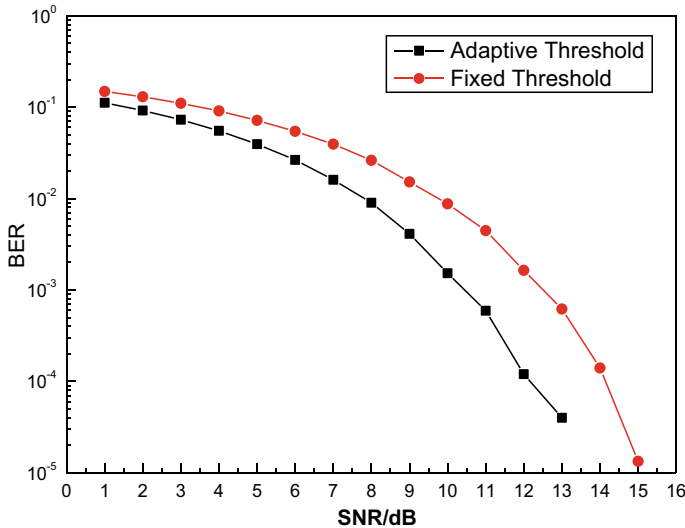


Fig. 7.14 Effect of judgment threshold on the BER [32]

The simulation results show that there is a significant reduction in the BER of the system after adopting the adaptive-judgment threshold, and a gain of approximately 1 dB can be achieved under the same BER condition.

7.3.7 Estimation-Error Analysis

The performance of the channel-estimation algorithm is characterized by the mean squared error (MSE), which indicates the dispersion of the estimated value with respect to the true value. Its mathematical expression is

$$MSE = E \left\{ \left(\hat{H} - H \right)^2 \right\}, \quad (7.91)$$

where $E(\cdot)$ represents the mean value, H the true value, and \hat{H} the value of each estimate. According to the parameter-estimation principle, when the MSE is smaller under the same estimation conditions, the estimation algorithm has better performance.

Adaptive-modulation coding relies on a knowledge of the channel-state information, and the choice of modulation or coding method is generally based on the received SNR of the channel; thus, channel-quality estimation is a prerequisite for adaptive-modulation coding. The estimation of the SNR is a problem of parameter estimation based on the received signal, and the most optimal estimate is one that is unbiased (or has minimum bias) and has minimum variance.

However, the actual obtained channel-quality estimate will have some error; therefore, the matching between the adaptive-modulation coding and the channel condition produces bias, which results in performance degradation. Here, the estimation error of the SNR is modeled as

$$\hat{SNR} = SNR + \Delta, \quad (7.92)$$

where SNR and \hat{SNR} are the true and estimated values of the channel SNR (in decibels), respectively, and Δ is a Gaussian random variable with variance σ^2 . The causes of this error include the following.

1. Noise and interference are additive and cannot be completely separated from the useful signal.
2. The time window for SNR estimation cannot be infinitely long, and the smaller the observation window, the larger the error.
3. Because the signal itself and atmospheric state change in real time, the estimation result will have a time delay relative to the signal transmission, which will also cause an estimation error.

The impact of an SNR-estimation error on the adaptive-coding system is manifested in two ways.

- 1. If $\hat{SNR} < SNR$, the adaptive-modulation coding algorithm will assign lower-order modulation and coding methods with a higher error-correction capability to the channel; thus, decreasing the information-transmission rate of the system.
- 2. If $\hat{SNR} > SNR$, higher-order modulation and poor error-correction coding are used, which increases the error probability of the system.

7.4 Experimental Measurement and Simulation

In optical wireless communication, the absorption, scattering, and turbulence effects of the atmosphere cause the laser spot to flicker, drift, and split, which significantly affects the stability and reliability of communication. Light-intensity fluctuations generally change according to the weather, and the overall trend is that light-intensity fluctuations are worse on sunny days than on cloudy or rainy days, with the smallest fluctuations when it rains. Rain not only reduces the ground temperature and abates the temperature difference, but also reduces the inhomogeneous components in the air.

To visually and accurately analyze the effect of random atmospheric channels on laser communication, we developed a real-time laser-measurement system using random atmospheric channels [40], for verifying the adaptability of laser-communication systems to various channels by obtaining experimental data, performing theoretical analysis and system modeling, and conducting meaningful exploration for further optimization of system-performance design.

We conducted experiments in cloudy, light-rain, medium-rain, heavy-rain, and foggy environments, based on the large amount of actual data measured by the atmospheric random-channel real-time measurement system described in Chap. 6. The experimental environmental parameters are presented in Table 7.1.

Table 7.1 Experimental environment parameters

Parameter	Value
Transmitting power	10 mW
Laser wavelength	650 nm
Modulation method	OOK
Transmission distance	100 m
Channel-estimation algorithm	Improved LS algorithm
SNR-estimation algorithm	Noise-variance-estimation based algorithm

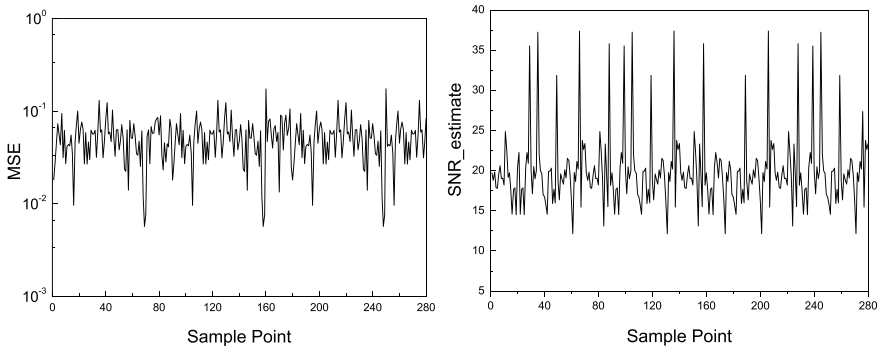


Fig. 7.15 Adaptive simulation results using real data of cloudy weather. **a** LS estimation. **b** SNR estimation [32]

Adaptive Simulation Under Cloudy Conditions

Figure 7.15a shows the estimated mean squared error of the LS algorithm under cloudy skies. Its SNR estimation is shown in Fig. 7.15b.

Adaptive Simulation Under Light-Rain Conditions

Figure 7.16a shows the estimated mean squared error of the LS algorithm under light-rain conditions; its SNR estimation is shown in Fig. 7.16b.

Adaptive Simulation Under Medium-Rain Conditions

Figure 7.17a shows the estimated mean squared error of the LS algorithm under medium-rain conditions; its SNR estimation is shown in Fig. 7.17b.

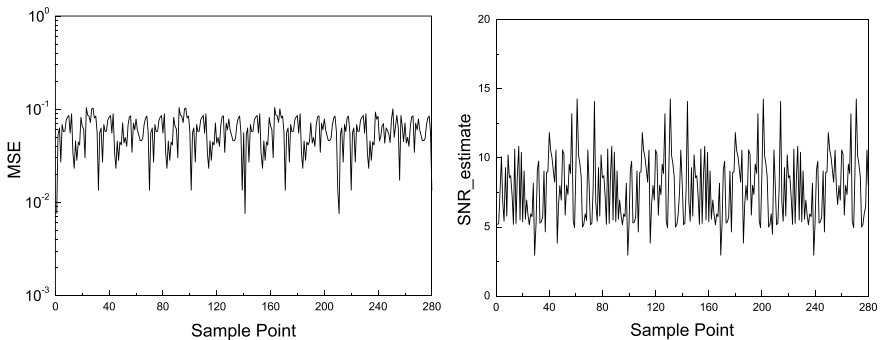


Fig. 7.16 Adaptive simulation results using real light-rain data. **a** LS estimation. **b** SNR estimation [32]

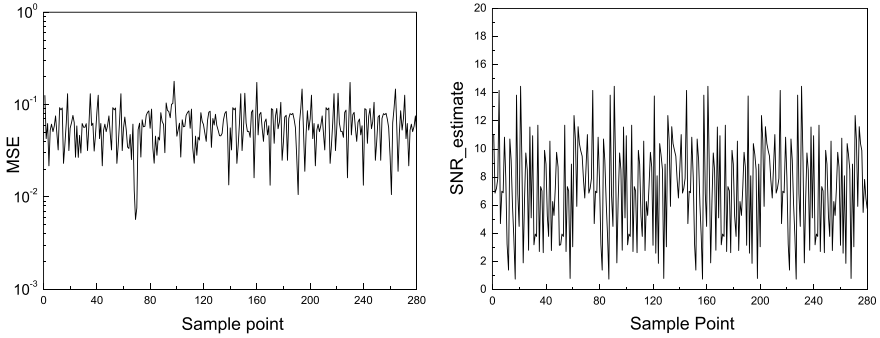


Fig. 7.17 Adaptive simulation results using real medium-rain data. **a** LS estimation. **b** SNR estimation [32]

Adaptive Simulation Under Heavy-Rain Conditions

The estimated mean squared error of the LS algorithm under heavy-rain conditions is shown in Fig. 7.18a, and its SNR estimation is shown in Fig. 7.18b.

Adaptive Simulation Under Foggy Conditions

The estimated mean squared error of the LS algorithm under foggy conditions is shown in Fig. 7.19a, and its SNR estimation is shown in Fig. 7.19b.

Integrated Adaptive Simulation for All Weather Conditions

It can be seen from the simulation results of various adaptive algorithms based on actual measurements that the mean squared error of channel estimation in cloudy, light, medium, and heavy-rain days is basically maintained at about 10^{-2} , and only on foggy days does the mean squared error increase significantly. It can be seen from the graph that the estimated value on foggy days will have a relatively large

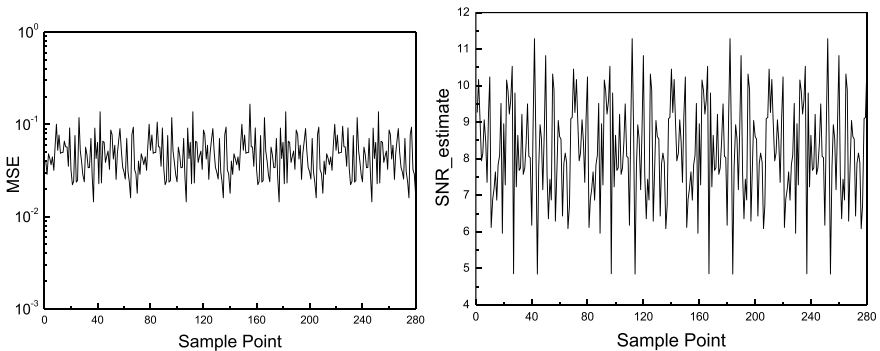


Fig. 7.18 Adaptive simulation results using real heavy-rain data. **a** LS estimation. **b** SNR estimation [32]

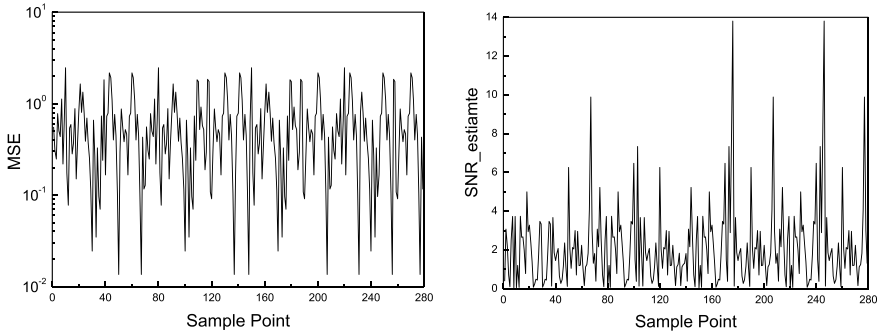


Fig. 7.19 Adaptive simulation results using real foggy-weather data. **a** LS estimation. **b** SNR estimation [32]

waveform distortion. This is likely due to the fact that the attenuation of the laser on foggy days is beyond the estimation capability. This is the most important effect of fog on the quality of wireless laser communications.

In Fig. 7.20, various weather variations are grouped for the simulation. A total of 280 sets of sampling points were used for the simulation, and four channel cases of cloudy, light rain, heavy rain, and foggy days were added; 70 sets of sampling points were used for each case. From the simulation results in Figs. 7.15, 7.16, 7.17, 7.18 and 7.19, it can be seen that the estimated received SNR for cloudy days is above 10 dB, the estimated received SNR for rainy days is between 4 and 12 dB, and the estimated received SNR for foggy days is below 5 dB.

In summary, referring to the simulation data, we can take 5 and 10 dB as the switching thresholds of adaptive-channel coding on cloudy days, although the atmospheric attenuation increases; however, owing to the reduction of atmospheric scintillation, we can use the same coding scheme as for sunny days. On rainy days, the effect of atmospheric scintillation is also relatively low, owing to the scattering effect of raindrops on the laser. This results in the optical power received by the receiver side having a relatively obvious reduction.

In foggy days, owing to the scattering of the laser by the suspended particles in the fog, and poor transmissions, the commonly used laser wavelengths are absorbed and attenuated very seriously; thus, the BER rises sharply, and it is necessary to consider the use of redundancy. A very high coding method or other alternate communication mode is needed to achieve the purpose of adaptive coding.

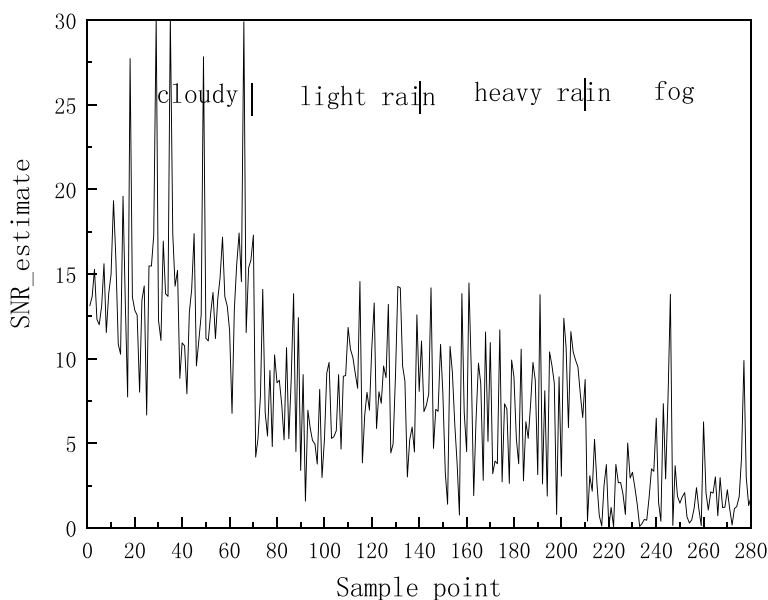


Fig. 7.20 Signal-to-noise ratio estimation in various weather conditions [32]

References

1. Ke X, Xi X (2004) Introduction to wireless laser communication. Beijing University of Posts and Telecommunications Press, Beijing, pp 45–69
2. Wang X (2006) Recent advances in foreign space laser communication system technology. *Telecommun Expr* 13(7):41–45
3. Ji W, Zhao C, Chen S (2002) Space optical communication system, technology, status and outlook. *Commun World* 09(6):30–34
4. Jia L, Lei J, Huang Z-M (2001) Composition of space laser communication system and its development. *Optoelectron Inf* 14(2):29–34
5. Pease R (1999) Optical laser communication systems carve niche in Metro markets. *Lightwave* 15(9):23–27
6. Vallestero NJ, Khushid M, Prasad NS et al (2002) Free-space optical communication systems (FOCUS): an Army overview. 48(21):276–282
7. Chan VJ, Bloom SH (1996) Results of 150-km, 1-Gbps lasercom validation experiment using aircraft motion simulator. 26(99):60–70
8. Zhao C, Ai Y, Tan Y (2005) Research on adaptive channel coding for spatial optical communication. *Space Electron Technol* 23(11):11–14
9. Fang X (2006) Research on adaptive coding modulation technology and its application in HSDPA. Harbin Institute of Technology, Harbin, pp 23–46
10. Yu J (2007) Research on wireless link adaptive technology. Beijing Jiaotong University, Beijing, pp 56–87
11. Wang Y, Yang G, Zhao Z (2005) Cybernetic analysis and design of adaptive coding strategy in optical wireless communication. *Opt Commun Technol* 16(5):11–14
12. Xie W, Tang J, Liu L (2004) Study of RCPT/ARQ adaptive error control mechanism in wireless optical communication. *J Electron* 32(2):223–226
13. Song X, Liu L, Dang A et al (2007) A new scheme of interleaved convolutional coding for high code rate wireless optical communication. *J Commun* 28(10):38–43

14. Fang X (2006) Research on adaptive coding modulation technique and its application in HSDPA. Harbin Engineering University, Harbin, pp 24–69
15. Zhou X, Song W, Li T (2002) Research on adaptive coding and modulation techniques in fading channels. *Commun Technol* 21(8):1–2
16. Zhang P, Han Y (2003) Estimation of signal-to-noise ratio using correlation coefficient. *J Military Eng* 24(1):136–138
17. Hu Q (2004) Atmospheric laser communication system channel research. Harbin Engineering University, Harbin, pp 31–45
18. Shu S, Yan J, Zheng Z et al (2007) Simulation of turbulent channel beam transmission in space laser communication. *J Light Scatter* 19(1):79–85
19. Wang L (2005) Numerical computation and simulation research for atmospheric laser communication. Xi'an University of Technology, Xi'an, pp 24–45
20. Ding D (2005) Design of PPM modulation and demodulation system for atmospheric laser communication. Xi'an University of Technology, Xi'an, pp 19–32
21. Pang Z, Park D, Zou C (2002) Performance comparison of several modulation methods in optical communication. *J Guilin Inst Electron Ind* 22(5):1–4
22. Proakis JG et al (2005) Modern communication systems (MATLAB edition), 2nd edn (trans: Liu S). Electronic Industry Press, Beijing, pp 36–45
23. Qiu T, Zhang X, Li X et al (2004) Statistical signal processing. Electronic Industry Publishing House, Beijing, pp 17–26
24. Yan D (2004) Research and simulation of channel estimation algorithm for MIMO system. Hohai University, Nanjing, pp 24–36
25. Van De Beek JJ, Edfors O, Sandell M, et al (1995) On channel estimation in OFDM systems. In: Countdown to the wireless twenty-first century, vol 2, no 12, pp 815–819
26. Hsieh MH, Wei CH (1998) Channel estimation for OFDM systems based on comb-type pilot arrangement in frequency selective fading channels. *IEEE Trans Consum Electron* 44(1):217–225
27. Zhang B (2005) Fundamental principles and key technologies of orthogonal frequency division multiplexing. National Defense Industry Press, Beijing, pp 121–136
28. Zhu Z (2005) Research on channel estimation techniques for OFDM-based systems. Southeast University, Nanjing, pp 25–36
29. Yang J, Xu Y, Zhao X et al (2006) Implementation of non-adaptive and adaptive channel estimation algorithms in DVB-T. *J Circuits Syst* 11(3):139–144
30. Edfors O, Sandell M, Van de Beek JJ et al (1998) OFDM channel estimation by singular value decomposition. *IEEE Trans Commun* 46(7):931–939
31. He J, Yanjun Hu (2007) Performance analysis of wireless optical MIMO system based on SVD algorithm. *J Anhui Univ* 31(3):34–37
32. Zhang Y (2008) Adaptive coding in atmospheric laser communication. Xi'an University of Technology, Xi'an, pp 30–35
33. Fan H, Chen J, Cao Z (2002) SNR estimation algorithm for non-constant envelope signals in AWGN channels. *J Electron* 30(9):1369–1371
34. Zhang X (1985) Modern signal processing. Tsinghua University Press, Beijing, pp 69–87
35. Pauluzzi DR, Beaulieu NC (2000) A comparison of SNR estimation techniques for the AWGN channel. *IEEE Trans Commun* 48(10):1681–1691
36. Omurcali C (2004) Channel estimation and its effect on the performance in an OFDM system, pp 23–41
37. Luo M (2007) Research on the estimation method of signal-to-noise ratio of OFDM system. Xi'an University of Electronic Science and Technology, Xi'an, pp 24–56
38. Wang H, Zhu Y (2006) Performance analysis of wireless optical communication modulation methods. *Adv Laser Optoelectron* 43(6):38–41
39. Fan C (2001) Principles of communication. National Defense Industry Press, Beijing, pp 97–121
40. Chen L (2005) Experimental measurement of atmospheric laser communication system. Xi'an University of Technology, Xi'an, pp 32–36

Chapter 8

Time-Domain Equalization Based on Adaptive Filtering



When a laser is transmitted through the atmosphere, its optical-pulse signal will be attenuated to varying degrees, and the pulse waveform will be wide, which may lead to the mutual interference of inter-frame signals. This will significantly affect the demodulation and decision decoding of the receiver, and needs to be overcome or minimized [1]. This chapter discusses the basic principles, applications, and algorithms of adaptive filtering [2]. The least mean squares (LMS) algorithm is studied, and the contradiction between the algorithm-convergence speed and the steady-state value of the mean squared error is discussed. To solve the contradiction caused by the fixed step size of the LMS algorithm, the variable-step-size LMS algorithm is deeply analyzed and studied.

8.1 Adaptive-Filtering Principle

By adding an adaptive-equalizer module to an atmospheric laser-communication system, the influence of the atmospheric channel on the transmission signal can be overcome. An adaptive equalizer is a typical application of an adaptive filter.

Time-domain equalizers can generally be divided into two categories [3–7]: linear and nonlinear. If the result of a decision in the receiver is fed back for equalizer-parameter adjustments, it is a nonlinear equalizer; otherwise, it is a linear equalizer. Figure 8.1 shows the basic equalizer-classification block diagram [5], which shows the structure and main algorithms of various equalizers.

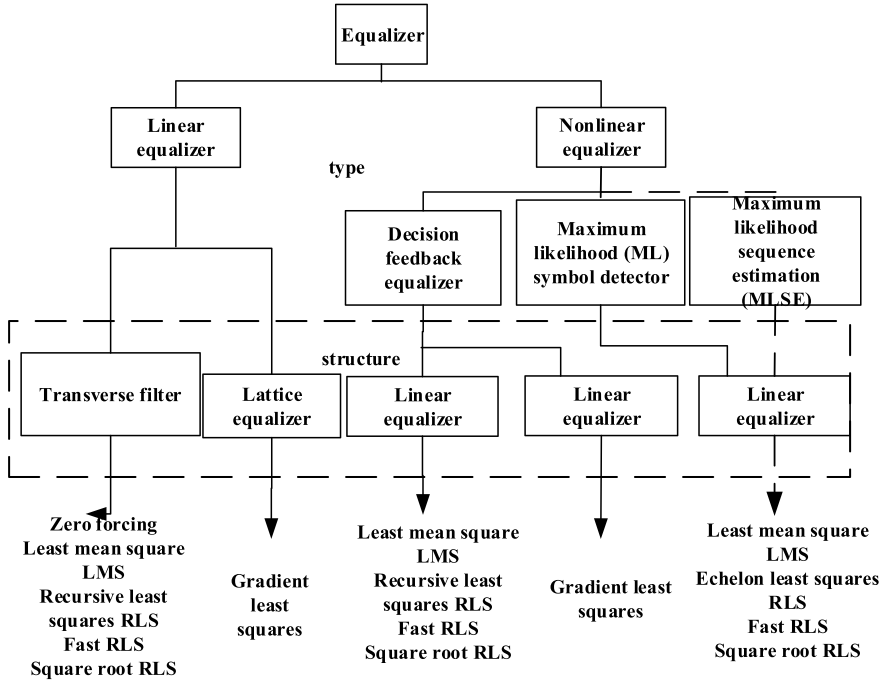


Fig. 8.1 Equalizer-structure classification block diagram

8.1.1 Adaptive Filters

When the statistical characteristics of an input process are unknown or change, an adaptive filter can adjust its parameters to meet the requirements of some optimal criterion. When the statistical characteristics of the input process are unknown, the process of an adaptive filter adjusting its own parameters is called a “learning process.” When the statistical characteristics of the input-process change, the process of an adaptive filter adjusting its own parameters is called a “tracking process” [2].

Figure 8.2 shows the general case of an adaptive filter. In the figure, $x(n)$ is the number of iterations, $y(n)$ represents the input signal, $d(n)$ represents the output signal of the adaptive filter and is the expected signal (reference signal). The error signal $e(n)$ is calculated by $d(n) - y(n)$. The error signal is used to construct the objective function required by an adaptive algorithm to determine the appropriate update mode for the filter coefficients. Minimizing the objective function means that the output signal of the adaptive filter matches the expected signal.

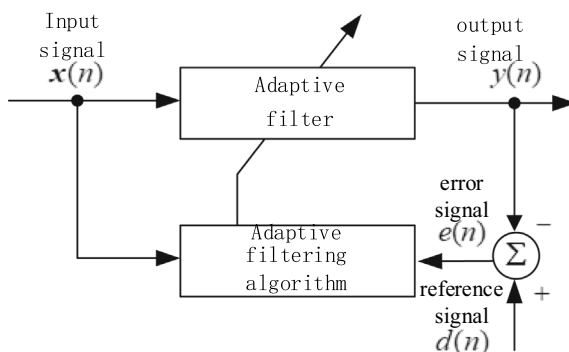


Fig. 8.2 Schematic diagram of an adaptive filter

8.1.2 Application of Adaptive Filters

Adaptive filters can work well in unknown environments, and have been widely used in communication, radar, sonar, earthquake, biomedical engineering, and other fields. The essential difference between the various adaptive-filter applications lies in the different methods of extracting the desired signal [1]. Adaptive-filter applications can be roughly divided into the following four categories [3–7].

(1) System identification

As shown in Fig. 8.3, in identification applications, adaptive filters are used to provide a linear model that best fits the unknown system in a sense. The system has the same input as the adaptive filter. The reference signal $d(n)$ of the adaptive filter is used as the system output. If the system has dynamic characteristics, the model provided by the adaptive filter is time-varying.

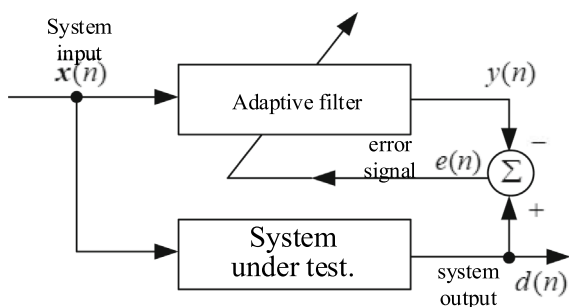
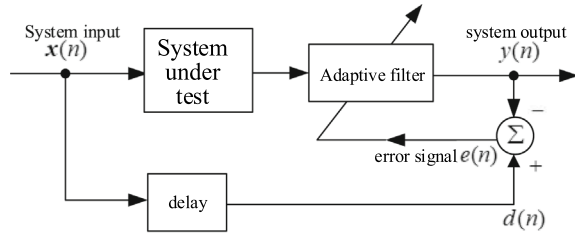


Fig. 8.3 System identification

Fig. 8.4 Equilibrium



In system-identification applications, given an unknown dynamic system, an adaptive filter is designed to approximate the dynamic system. Applications in layered-earth modeling and seismic exploration develop and study the layered model of the earth to explain complex problems on the earth's surface [1].

(2) Inverse model (equilibrium)

As shown in Fig. 8.4, one function of an adaptive filter is to provide an inverse model. In the case of a linear system, the transfer function of the inverse model is equal to the reciprocal of the transfer function of the unknown system. The input delay of the system constitutes the reference signal $d(n)$ of the adaptive filter.

Given a channel with an unknown impulse response, the purpose of an adaptive equalizer is to control the channel output so that the channel and equalizer cascade provides a good approximation of the ideal transmission medium.

(3) Forecasting

As shown in Fig. 8.5, the adaptive filter can predict the current value of a random signal. The current value of signal $d(n)$ is used as the reference signal of the adaptive filter. The past value of the signal is added to the input of the filter. The output or estimation error of the adaptive filter can be used as the output of the system. In the former case, the system acts as a predictor; in the latter case, the system acts as a prediction-error filter.

In predictive-coding applications, adaptive prediction is used to develop the model of the signal of interest (such as a speech signal), rather than directly coding the signal. In this encoding, the prediction error is encoded for transmission and storage. Typically, the prediction error has a smaller variance than the original signal, which serves as the basis for improving the coding.

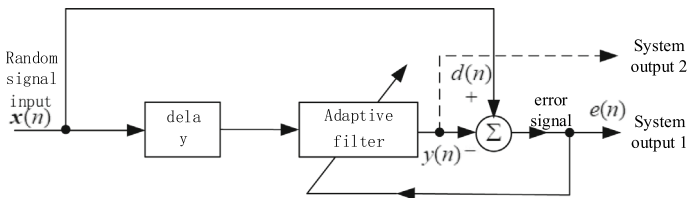


Fig. 8.5 Forecasting

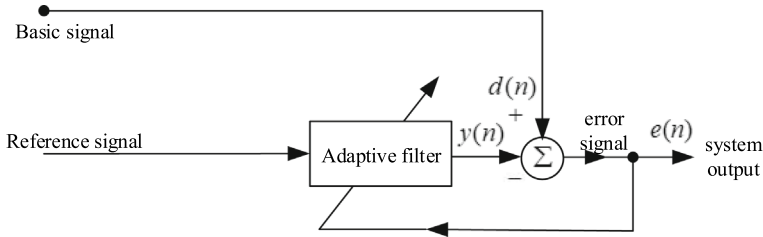


Fig. 8.6 Interference cancellation

(4) Interference cancellation

As shown in Fig. 8.6, an adaptive filter is used to eliminate the unknown interference contained in a basic signal. The basic signal is used as the desired response of the adaptive filter, and the reference signal is used as the input of the filter. The reference signal comes from a sensor or a group of sensors.

In noise-cancellation applications, the purpose of an adaptive noise canceller is to remove noise from the received signal to improve the signal-to-noise ratio (SNR). The echo cancellation encountered in a telephone circuit is a special form of noise cancellation.

8.1.3 Adaptive-Filtering Algorithm

The optimization criterion of an adaptive-filtering algorithm is to minimize the objective function. The objective function F is a function of $x(n)$, $d(n)$, $y(n)$, and $F = F[x(n), d(n), y(n)]$, for which $x(n)$, $d(n)$, $y(n)$ are the input signal, reference signal, and output signal of the adaptive filter, respectively. This objective function satisfies the nonnegativity criterion: for any $x(n)$, $d(n)$, $y(n)$, there exists $F = F[x(n), d(n), y(n)] \geq 0$ [3].

In the adaptive-filtering-algorithm operation process, the adaptive algorithm minimizes the function, so that $y(n)$ is approximately equal to the filter $d(n)$, and the parameters of the filter converge to the set of optimal coefficients that minimize the objective function. The three basic elements of the adaptive algorithm are as follows:

1. Definition of common objective functions, $F[e(n)]$:

$$\text{Mean square error (MSE): } F[e(n)] = E[|e(n)|^2] \quad (8.1a)$$

$$\text{Least squares (LS) : } F[e(n)] = \frac{1}{n+1} \sum_{i=0}^n |e(n-i)|^2 \quad (8.1b)$$

$$\text{Weighted least squares (WLS) : } F[e(n)] = \sum_{i=0}^n \lambda^i |e(n-i)|^2 \quad (8.1c)$$

where λ is a constant less than 1;

Instantaneous square value (ISV):

$$F[e(n)] = |e(n)|^2 \quad (8.1d)$$

Generally speaking, ISV is easier to implement; however, because its objective function is extremely simplified, it has a noise-convergence problem.

2 Objective-function F minimization algorithm

Steepest-descent method: This method searches for the minimum point of the objective function in the direction opposite the gradient vector of the objective function. This method is also called a gradient method.

3. Definition of error signal $e(n)$

The selection of the error signal is generally determined by the reference signal and output signal, according to the needs of the actual equalizer.

(1) LMS algorithm

Widrow and Hoff proposed the least mean squares (LMS) algorithm in 1960 [2]; it does not need to calculate the relevant correlation function or matrix inversion. The LMS algorithm is widely used in adaptive filtering because of its simplicity, low calculation, and easy real-time processing. The iterative formula of the LMS algorithm based on the steepest-descent method is as follows:

$$e(n) = d(n) - x^T(n)w(n) \quad (8.2)$$

$$w(n+1) = w(n) + 2\mu e(n)x(n) \quad (8.3)$$

where $w(n)$ is the weight vector of the adaptive filter at time n , $x(n) = [x(n), x(n-1), \dots, x(n-L+1)]^T$ is the input signal vector of the time- n filter, and L is the order of the adaptive filter. $d(n)$ is the expected signal, the error signal is $e(n)$, and the step factor is μ . The convergence condition of the LMS algorithm is $0 < \mu < \frac{1}{\lambda_{\max}}$, in which λ_{\max} is the maximum eigenvalue of the autocorrelation matrix of the input signal.

The convergence speed, tracking speed, and steady-state value of the mean squared error are the three most important technical indexes to measure the advantages and disadvantages of the adaptive-filtering algorithm. The number of iterations until the mean squared error of the adaptive filtering algorithm reaches convergence is the convergence speed of the algorithm. The smaller the number of iterations, the faster the convergence speed, and the better the algorithm performance.

When the channel characteristics change, the number of iterations required for the mean squared error of the adaptive-filtering algorithm to reach convergence is called the tracking speed of the algorithm. The fewer the number of iterations, the faster the tracking speed, and the better the algorithm performance. The mean squared error

after the adaptive-filtering algorithm convergence is called the steady-state value of the mean squared error. The smaller the steady-state value of the mean squared error, the higher the convergence accuracy, and the better the algorithm performance.

Reducing the step factor μ can reduce the steady-state value of the mean squared error of the adaptive-filtering algorithm and improve its convergence accuracy. However, this will also reduce the convergence and tracking speeds of the algorithm. Therefore, the requirements of a fixed-step-size adaptive-filtering algorithm for adjusting the step-size factor μ are contradictory in terms of the convergence speed, time-varying system-tracking speed, and convergence accuracy. To overcome this contradiction, many variable-step-size adaptive-filtering algorithms have been proposed.

(2) Recursive least squares (RLS) algorithm

The least squares (LS) algorithm directly processes the received data to minimize the objective function. The recursive least squares (RLS) algorithm is most used in practice and its iterative formula is as follows:

$$\mu(n) = x^T(n)C(n-1)x(n) \quad (8.4)$$

$$g(n) = \frac{C(n-1)x(n)}{\lambda + \mu(n)} \quad (8.5)$$

$$w(n) = w(n-1) + g(n)[d(n) - x^T(n)w(n-1)] \quad (8.6)$$

$$C(n) = \lambda^{-1}[C(n-1) - g(n)x^T(n)C(n-1)] \quad (8.7)$$

Among them, $C(0) = \delta^{-1}I$, and the δ parameter setting is related to the SNR. A small value is taken when the SNR is high, and a large value is taken when the SNR is low. I is the unit matrix, and parameter λ is the exponential weighting factor, which meets $0 < \lambda < 1$ [1].

The RLS algorithm updates the inverse matrix of the autocorrelation matrix of the input signal by recursive estimation. It has a fast convergence speed, and its convergence performance is independent of the spectral characteristics of the input signal. However, the least-squares algorithm has two disadvantages. One is its complexity. The number of calculations for one iteration of the algorithm is directly proportional to N^2 , which requires a large amount of calculation and a large amount of storage, which are not conducive to real-time signal processing. Second, if the inverse matrix of the estimated autocorrelation matrix loses the positive definite characteristic, it will cause the algorithm to diverge.

(3) Transform-domain adaptive-filtering algorithm

For an input signal with strong correlation, after some orthogonal transformations, the eigenvalue divergence of the autocorrelation matrix of the input signal will be reduced. The smaller the eigenvalue divergence of the input-signal autocorrelation

matrix, the better the convergence performance of the LMS algorithm. The general steps of the transform-domain adaptive-filtering algorithm are as follows:

1. An orthogonal transform is selected to transform the time-domain signal into a transform-domain signal;
2. The transformed signal is normalized by the square root of its energy;
3. A self-adaptive algorithm is used for filtering.

A wavelet transform is also used in transform-domain adaptive filtering. In wavelet-transform adaptive filtering, two forms are usually adopted: one is wavelet-sub-band adaptive filtering, which is equivalent to transforming the input signal and the expected signal into a time-domain output signal after adaptive filtering in the multi-resolution space. The second is adaptive filtering in a wavelet-transform domain. The input signal is represented by the signal in the wavelet multi-resolution space, while the expected response signal is not transformed by the wavelet [4].

(4) Affine-projection algorithm

The affine-projection algorithm is the generalization of the normalized least mean squares (NLMS) algorithm. Its performance is between that of the LMS algorithm and the RLS algorithm, and its computational complexity is lower than that of the RLS algorithm. In the fast affine-projection algorithm, the sliding-window fast-transverse filter algorithm is used to calculate the pre-filter vector, which avoids the matrix-inversion operation. Although the computational complexity of the fast affine-projection algorithm is reduced, the implementation of the embedded sliding-window fast-transverse filter algorithm is relatively complex, and it has a problem with numerical stability [4].

(5) Other adaptive-filtering algorithms

In addition to the above adaptive-filtering algorithms, other algorithms include the conjugate-gradient algorithm, adaptive-filtering algorithm based on QR decomposition [3], periodic LMS algorithm, max NLMS algorithm, n -max NLMS algorithm, LMF (RLF) algorithm, and leaky LMS algorithm [4].

(6) Adaptive-filtering algorithm selection criteria [7]

1. Convergence speed: The number of iterations required for the algorithm to reach the best statistical-approximation value when the input is a stationary signal.
2. Offset: The difference between the output of the adaptive algorithm after convergence and the theoretical value.
3. Tracking characteristics: The adaptive algorithm is required to track environmental changes in a non-stationary environment. The tracking characteristic is restricted by two parameters: the convergence speed and steady-state wave momentum.
4. Robustness: If the system is robust, a small disturbance of the adaptive system by internal or external factors will only cause a small estimation error.

5. Computation: This includes the computation required to complete an iteration, the storage space required to store data and programs, and the programming workload to realize the algorithm.

Three criteria are important for selecting adaptive algorithms: computational complexity, convergence performance, and robustness.

8.2 Analysis of LMS-Like Algorithms

8.2.1 LMS-Algorithm Analysis

(1) LMS algorithm

The steepest-descent method is the basis of the least mean squares algorithm. It seeks the best value of the weighted vector by recursion and constitutes the basis of many algorithms, especially the most widely used LMS algorithm [3]. The recursive formula of the steepest-descent method is as follows:

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu \hat{\nabla}_{\mathbf{w}} \xi \quad (8.8)$$

$$\mathbf{w}(n+1) - \mathbf{w}_{opt} = (\mathbf{I} - 2\mu R_{xx})[\mathbf{w}(n) - \mathbf{w}_{opt}] \quad (8.9)$$

where $\xi = E\{|e(n)|^2\}$ is the objective function of the algorithm, $\hat{\nabla}_{\mathbf{w}} \xi$ is the estimated value of the gradient, \mathbf{w}_{opt} is the best weight vector, \mathbf{I} is the unit matrix, and R_{xx} is the autocorrelation matrix of the input vector. The gradient-estimation value of the LMS algorithm is

$$\hat{\nabla}_{\mathbf{w}} \xi = \hat{\nabla}_{\mathbf{w}} E\{|e(n)|^2\} = \nabla_{\mathbf{w}} |e(n)|^2 \quad (8.10)$$

The gradient $\nabla_{\mathbf{w}} |e(n)|^2$ of the instantaneous error power is used as the estimated value of the mean squared error gradient, $\nabla_{\mathbf{w}} E\{|e(n)|^2\}$. That is, the instantaneous squared error performance function $|e(n)|^2$ is used to replace the mean squared error performance function:

$$\xi = E\{|e(n)|^2\} \quad (8.11)$$

We substitute Eq. (8.10) into Eq. (8.8):

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu \nabla_{\mathbf{w}} |e(n)|^2 \quad (8.12)$$

$$e(n) = d(n) - y(n) = d(n) - \mathbf{w}^T(n)\mathbf{x}(n) \quad (8.13)$$

to obtain

$$\nabla_{\mathbf{w}} |e(n)|^2 = -2e(n)\mathbf{x}(n) \quad (8.14)$$

We substitute Eq. (8.14) into Eq. (8.12) to obtain the recursive LMS algorithm formula:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n) \quad (8.15)$$

Figure 8.7 shows the implementation of the LMS algorithm. Each iteration of the LMS algorithm requires multiplication to update the filter coefficients and to generate error signals. The iterative formula of the LMS algorithm is as follows:

$$y(n) = \mathbf{w}^T(n)\mathbf{x}(n) \quad (8.16)$$

$$e(n) = d(n) - y(n) \quad (8.17)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n) \quad (8.18)$$

where $\mathbf{w}(n)$ is the tap weight vector of the adaptive filter.

(2) Selection of the LMS-algorithm convergence step

Research on the LMS algorithm mainly includes its convergence and its steady-state offset performance. The step size μ plays a key role in these two aspects. The size of μ determines the convergence performance of the algorithm [5, 6]. The upper limit of the convergence step of the LMS algorithm given in the literature [5, 6] is

$$\mu_{av} = \frac{DM(1+2DM)}{(1+DM)(1+3DM)tr\mathbf{R}} \quad (8.19)$$

where DM is the given offset and $tr\mathbf{R}$ is the trace of the autocorrelation matrix of the input vector.

(3) Performance analysis of the LMS algorithm

The LMS algorithm has the advantages of a simple structure and strong robustness; however, its disadvantage is a slow convergence speed. This must be addressed to achieve real-time processing. The fundamental reasons for the slow convergence speed of LMS algorithm are as follows [8–12].

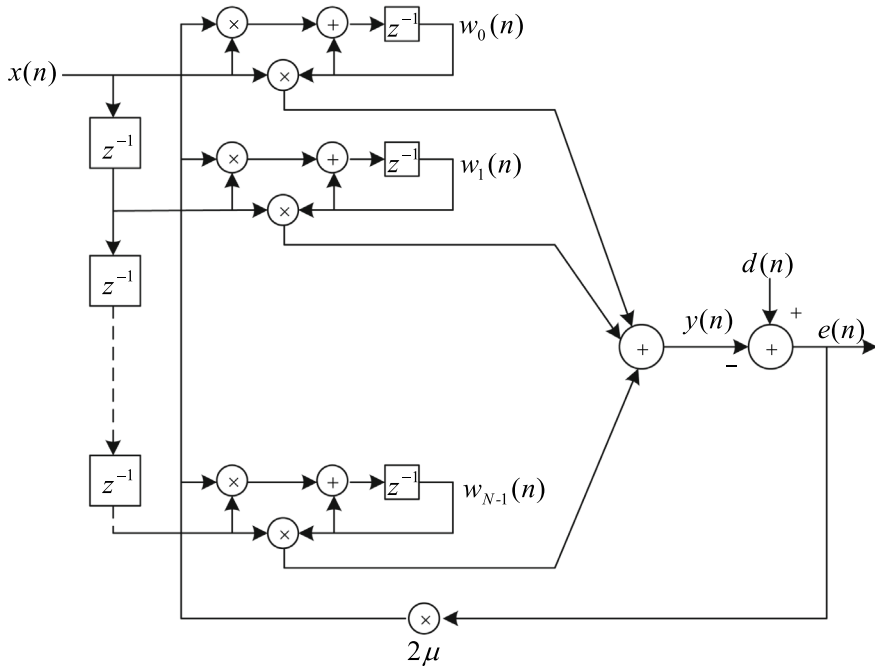


Fig. 8.7 LMS adaptive finite-impulse-response (FIR) filter

1. The step size cannot be larger; otherwise the algorithm will not converge.
2. The steady-state values of a fast convergence and a small mean squared error cannot be obtained simultaneously.

These reasons are related to the step size. The step size in the LMS algorithm is the only parameter that can control the iterative process; therefore, it must be the starting point to improve the algorithm's performance. The value range of the convergence step can be increased using a variable step factor. This method will be described in detail in the following sections.

8.2.2 Normalized LMS Algorithm

The normalized LMS (NLMS) algorithm can reduce the eigenvalue divergence of the autocorrelation matrix of the input signal; that is, reduce the correlation of the input signal and increase the value range of the convergence step [3]. A variable convergence factor μ_n is introduced into the renewal equation of the standard LMS algorithm. At this time, the renewal equation of the tap weight vector can be expressed as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu_n \mathbf{x}(n)e(n) \quad (8.20)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \Delta \tilde{\mathbf{w}}(n) \quad (8.21)$$

Among them, $\Delta \tilde{\mathbf{w}}(n)$ is the variation of the tap weight vector, and the selection of μ_n is aimed at achieving a faster convergence. The following strategy for reducing the instantaneous squared error is adopted. The instantaneous squared error is expressed as follows:

$$\begin{aligned} e^2(n) &= [d(n) - \mathbf{w}^T(n)\mathbf{x}(n)]^2 \\ &= d^2(n) + \mathbf{w}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n) - 2d(n)\mathbf{w}^T(n)\mathbf{x}(n) \end{aligned} \quad (8.22)$$

The update change of the weight vector can be given by the following formula:

$$\tilde{\mathbf{w}}(n) = \mathbf{w}(n) + \Delta \tilde{\mathbf{w}}(n) \quad (8.23)$$

Then, the corresponding squared error is

$$\begin{aligned} \tilde{e}^2(n) &= e^2(n) + \Delta \tilde{\mathbf{w}}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n) + \mathbf{w}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\Delta \tilde{\mathbf{w}}(n) \\ &\quad - 2d(n)\Delta \tilde{\mathbf{w}}^T(n)\mathbf{x}(n) + \Delta \tilde{\mathbf{w}}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\Delta \tilde{\mathbf{w}}(n) \\ &= e^2(n) - 2\Delta \tilde{\mathbf{w}}^T(n)\mathbf{x}(n)e(n) + \Delta \tilde{\mathbf{w}}^T(n)\mathbf{x}(n)\mathbf{x}^T(n)\Delta \tilde{\mathbf{w}}(n) \end{aligned} \quad (8.24)$$

$$\begin{aligned} \Delta e^2(n) &= \tilde{e}^2(n) - e^2(n) \\ &= -4\mu_n e^2(n)\mathbf{x}^T(n)\mathbf{x}(n) + 4\mu_n^2 e^2(n)[\mathbf{x}^T(n)\mathbf{x}(n)]^2 \end{aligned} \quad (8.25)$$

To improve the convergence speed, $\Delta e^2(n)$ is minimized.

$$\frac{\Delta e^2(n)}{\partial \mu_n} = 0 \quad (8.26)$$

Then, we obtain

$$\mu_n = \frac{1}{2\mathbf{x}^T(n)\mathbf{x}(n)} = \frac{1}{2\|\mathbf{x}(n)\|^2} \quad (8.27)$$

Now,

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\mathbf{x}(n)e(n)}{\|\mathbf{x}(n)\|^2} \quad (8.28)$$

To control the imbalance, a fixed convergence factor μ is introduced. To avoid a large step size when $\|\mathbf{x}(n)\|^2$ is very small, the γ parameter is introduced. Therefore,

updating the equation of the tap weight vector produces

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\mu}{\gamma + \|\mathbf{x}(n)\|^2} \mathbf{x}(n)e(n) \quad (8.29)$$

In Eq. (8.29), the tap input vector $\mathbf{x}(n)$ is normalized. The value of μ is determined by Eq. (8.19) and γ is a small normal number.

8.2.3 Simplified LMS Algorithm

A recursive formula from the standard LMS algorithm is designed:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu\mathbf{x}(n)e(n) \quad (8.30)$$

It can be seen that the adaptive-adjustment direction of the weight vector depends only on the symbol of the weight vector $\mathbf{x}(n)e(n)$; thus, Eq. (8.3) can be simplified into the following forms:

$$\text{Simplified algorithm 1 : } \mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu\text{sign}[\mathbf{x}(n)]e(n) \quad (8.31)$$

$$\text{Simplified algorithm 2 : } \mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu\mathbf{x}(n)\text{sign}[e(n)] \quad (8.32)$$

$$\text{Simplified algorithm 3: } \mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu\text{sign}[\mathbf{x}(n)][e(n)] \quad (8.33)$$

where

$$\text{sign}(x) = \begin{cases} \frac{x}{|x|} & x \neq 0 \\ 0 & x = 0 \end{cases} \quad (8.34)$$

Substituting Eq. (8.34) into Eqs. (8.31), (8.32), and (8.33), three simplified algorithmic expressions can be obtained:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\frac{\mu}{|\mathbf{x}(n)|} \mathbf{x}(n)e(n) \quad (8.35)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\frac{\mu}{|e(n)|} \mathbf{x}(n)e(n) \quad (8.36)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\frac{\mu}{|\mathbf{x}(n)||e(n)|} \mathbf{x}(n)e(n) \quad (8.37)$$

From algorithmic expressions (8.35), (8.36), and (8.37), it can be seen that the above three algorithms can be regarded as variable-step-size LMS algorithms. At the

beginning of the algorithmic-convergence process, the initial convergence processes of algorithms 2 and 3 are very slow, while algorithm 1 does not have this disadvantage. The processing of algorithm 3 is the simplest. If a power of 2 is selected, there is no need for a multiplier for algorithms 2 and 3. A set of shift registers and adders can be used to update the weight vector. The advantage of the simple operation of this algorithm is that it can be used for a reference in the practical application of other variable-step-size LMS algorithms [7].

8.2.4 Variable-Step-Size LMS Algorithm [10–28]

The traditional LMS algorithm μ has a fixed step size, and there is a contradiction between the convergence speed, tracking ability, and steady-state value of the mean squared error. A small step size μ ensures a small offset in the steady state; however, the convergence speed will be slow and the tracking ability will be poor. On the other hand, a large step size μ gives the algorithm a fast convergence speed and good tracking ability, at the cost of a large misalignment. Using a variable-step-size factor is an effective method to solve the contradiction caused by the fixed step size in the LMS algorithm.

In view of the shortcomings of the LMS algorithm, a variable step size to adjust the weight vector is as follows: When the weight-vector coefficient is far from the optimal weight-vector coefficient, we select a larger step size to improve the convergence and tracking speeds. When the error between the output signal and the expected signal is reduced, the weight-vector coefficient is close to optimal and a smaller step is selected to obtain the steady-state value of a smaller mean squared error.

The variable-step-size LMS algorithm uses some approximations provided in the adaptive process to adjust the step size. A relatively simple and effective method is to use the error signal in the adaptive process to establish a functional relationship between the step size and the error signal:

1. Establish a relationship between the step size and error signal;
2. Establish a relationship between the step size, $e(n)$, and the estimated value of the cross-correlation function $e(n)$ of the sum;
3. The step size is related to the autocorrelation estimation of the error signal.

(1) Relationship between the step size and error signal

a. LE-LMS algorithm

Reference [8] proposed a linear exponential (LE) adaptive-filtering algorithm, based on the optimal estimation of the gradient of the steepest-descent method. The iterative formula of the algorithm is as follows:

$$\mathbf{w}(n+1) = \begin{cases} \mathbf{w}(n) + 2\mu_{\max}e(n)\mathbf{x}(n) \left[1 - \exp\left(-\ln\left(\frac{e^2(n)}{\delta_n^2} - 1\right)\right) \right], & e^2(n) \geq 2\delta_n^2 \\ \mathbf{w}(n) + 2\mu_{\min}e(n)\mathbf{x}(n), & e^2(n) < 2\delta_n^2 \end{cases} \quad (8.38)$$

where $\delta_n^2 = E\{e^2(n)\}$

μ_{\max} is determined by the upper bound of the algorithm convergence with μ to ensure that the weight vector can quickly approach the optimal weight-vector coefficient at the beginning. When the weight vector is close to the optimal weight-vector coefficient, that is, $e^2(n) < 2\delta_n^2$, the step size of the algorithm μ_{\min} is selected to ensure that the algorithm has a small steady-state value of the mean squared error in the convergence stage. μ_{\min} is determined by the amount of required misadjustment: $DM = \frac{N\mu\delta_n^2}{2 \cdot N\mu\delta_n^2}$ [8].

This algorithm optimizes the gradient-estimation value and processes the error signal nonlinearly. Although this algorithm has an excessive amount of computation, the nonlinear processing idea is a good reference for the variable-step-size algorithm.

b. SVS-LMS algorithm

Similar to the LE-LMS algorithm, Ref. [9] uses the nonlinear processing idea and the absolute value of the error signal to control the change of the step size; hence, the step size is relatively large in the initial convergence stage. It can also be used when the parameters of a time-varying system change, to have a faster convergence and tracking speed of the time-varying system. After the algorithm converges, a small step size is used to achieve the steady-state value of a small mean squared error. The step size of the algorithm μ proposed in Ref. [9] is determined by the *sigmoid* function of $e(n)$:

$$\mu(n) = \beta \left(\frac{1}{1 + \exp(-\alpha|e(n)|)} - 0.5 \right) \quad (8.39)$$

where α is the constant of the control S-function shape and β is the constant of the control S-function range. The function curve with $\mu(n)$ and $e(n)$ is shown in Fig. 8.8. When $e(n)$ is large, $\mu(n)$ is also large; however, $\mu(n)$ will not exceed the limit of $\beta/2$. β is a constant that needs to be determined empirically. The optimal value of the rise speed of curve α also needs to be determined empirically.

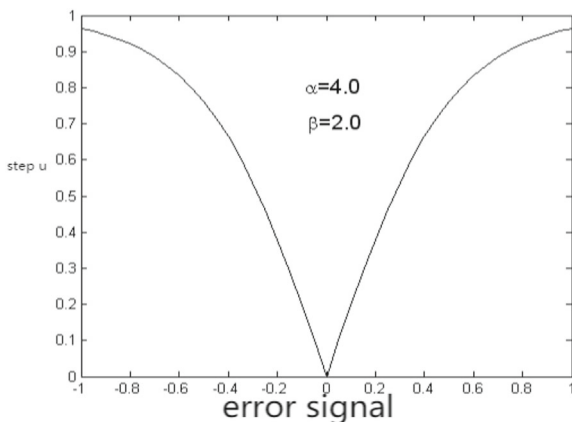
The iterative formula of the S-function variable-step-size LMS (SVS-LMS) algorithm is as follows:

$$e(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n) \quad (8.40)$$

$$\mu(n) = \beta \left(\frac{1}{1 + \exp(-\alpha|e(n)|)} - 0.5 \right) \quad (8.41)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu(n)e(n)\mathbf{x}(n) \quad (8.42)$$

Fig. 8.8 Function curve of step lengths $\mu(n)$ and $e(n)$



The difference between this algorithm and the LMS algorithm is that μ can vary. When the algorithm $|e(n)|$ enters a steady state, $\mu(n)$ reaches its minimum value, and the weight-vector coefficient $\mathbf{w}(n)$ approaches its optimal value.

In the SVS-LMS algorithm, the limit $\beta/2$ will not be exceeded by $\mu(n)$; thus, the maximum β value is $\beta_{max} = 2\mu^*$, where μ^* is determined by Eq. (8.18). Because $\mu(n)$ is variable and can achieve great results in the initial and tracking stages, the SVS-LMS algorithm has faster convergence and tracking speeds than the fixed-step LMS algorithm. When the algorithm enters the convergence steady state, $e(n)$ is very small. At this time, $\mu(n) \approx 0$ and the steady-state value of the mean squared error decreases with the decrease of the step size. Therefore, the SVS-LMS algorithm has a smaller steady-state mean squared error value than the LMS algorithm.

c. VS-LMS-E-1 algorithm

The SVS-LMS algorithm discussed above introduces the *sigmoid* function to adjust the step size of the adaptive algorithm, and obtains a faster convergence speed and a smaller steady-state value of the mean squared error. However, the *sigmoid* function is too complex, and changes rapidly when the error $e(n)$ is close to zero; hence, the SVS-LMS algorithm still has a large step-size in the steady-state stage. Reference [10] provides another function that satisfies the principle of variable-step-size adjustment:

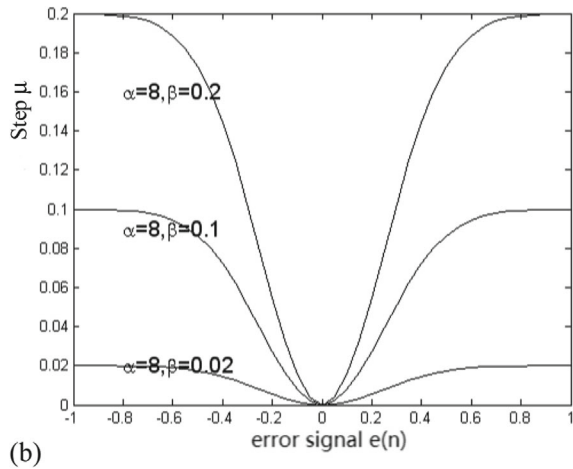
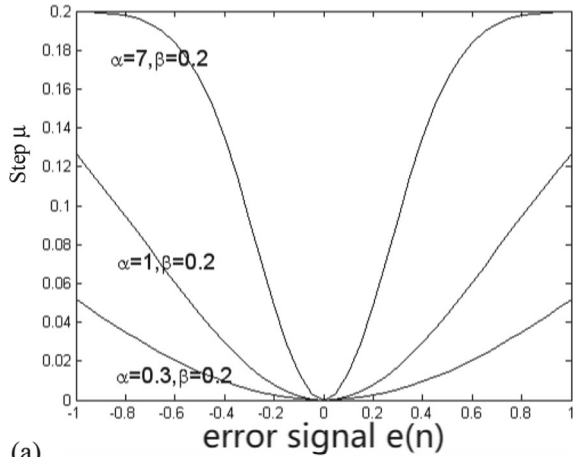
$$\mu(n) = \beta(1 - \exp(-\alpha|e(n)|^2)) \quad (8.43)$$

where $\alpha > 0$ controls the shape of the parameter and $\beta > 0$ controls the value range of the parameter. The function curve with step length $\mu(n)$ and $e(n)$ is shown in Fig. 8.9a and b.

The iterative formula of the VS-LMS-E-1 algorithm is as follows:

$$e(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n) \quad (8.44)$$

Fig. 8.9 Function curves when α and β change
a Different α with the same β **b** Different β with the same α



$$\mu(n) = \beta(1 - \exp(-\alpha|e(n)|^2)) \quad (8.45)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu(n)e(n)\mathbf{x}(n) \quad (8.46)$$

As can be seen from Fig. 8.9a and b, the step size $\mu(n)$ changes gradually when the error $e(n)$ is close to zero, which overcomes the deficiency of the *sigmoid* function in the step-size adjustment in the steady-state stage. In the initial stage, the correspondence $e(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n)$ is large, and the convergence speed of the algorithm $\mu(n)$ is fast. When $|e(n)|$ enters a steady state, $\mu(n)$ becomes increasingly smaller, and the filter coefficient approaches the optimal value.

The values of α and β can be selected according to the size of the initial error value $|e(n)|$, so that its corresponding value is as large as possible when satisfying the convergence of $\mu(n)$, to expedite the convergence speed.

The algorithm overcomes the step-size adjustment deficiency of the SVS-LMS algorithm in the steady-state stage, and the calculation amount is relatively small.

d. VS-LMS-E-2 algorithm

In the VS-LMS-E-1 algorithm proposed above, the step size $\mu(n)$ changes slowly when the error $e(n)$ is close to zero; however, this is at the cost of reducing the step value, especially when the error is large. To make the step size change slowly when the error is close to zero, the value of α must be reduced, and the step value will be reduced under the same value β . To solve this problem, Refs. [11, 12] proposed an improved algorithm. The new algorithm introduces a compensation factor γ . Its algorithmic iteration formula is as follows:

$$e(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n) \quad (8.47)$$

$$\mu(n) = \beta(1 - \exp(-\alpha|e(n)|^\gamma)) \quad (8.48)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu(n)e(n)\mathbf{x}(n) \quad (8.49)$$

where β is the value range of the control function, while α and γ control the shape of the function. and γ is called the compensation factor. The function curve of step size $\mu(n)$ and error $e(n)$ is shown in Fig. 8.10a and b.

As can be seen from Fig. 8.10, according to the actual system requirements, through the appropriate selection of α , β , and γ , the curve can be constructed in line with the variable-step-size adjustment principle.

e. DVS-LMS algorithm

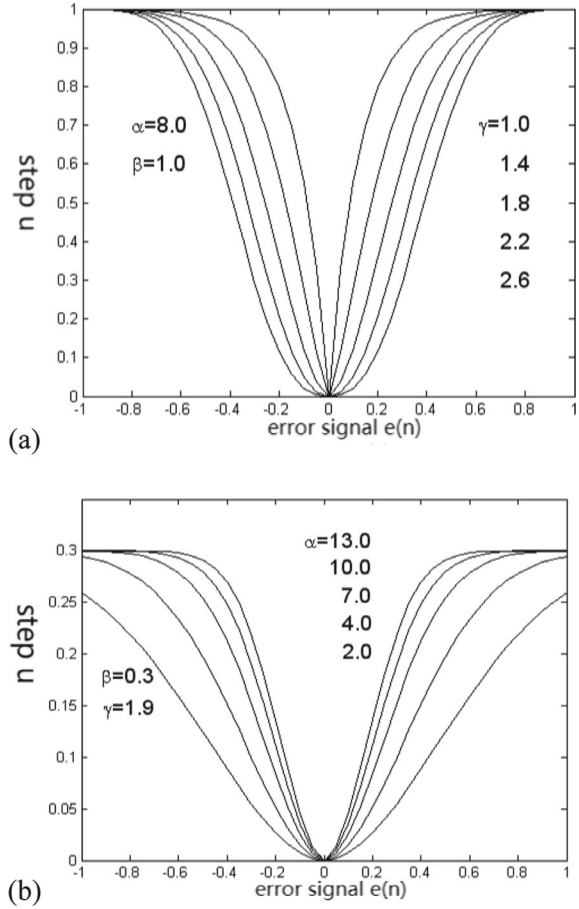
The SVS-LMS algorithm has been improved several times. Although it overcomes the defect that $|e(n)|$ cannot change slowly near zero, it still requires an exponential operation, which uses a large amount of calculation, especially in a hardware implementation. Reference [13] proposed a variable-step-size LMS algorithm based on a skip-tongue line. The step size is large in the initial stage; however, the step size factor is small and changes little in the steady-state stage. In addition, there is no need for an exponential operation, which greatly reduces the computational complexity [13].

The iterative formula of the variable-step-size LMS algorithm based on a skip-tongue line is as follows:

$$e(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n) \quad (8.50)$$

$$\mu(n) = \beta \left[1 - \frac{1}{\alpha \times e^2(n) + 1} \right] \quad (8.51)$$

Fig. 8.10 Function curve when γ and α change (a) α and β are the same, with different γ function curves (b) γ and β are the same, with different α function curves



$$\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu(n)e(n)\mathbf{x}(n) \quad (8.52)$$

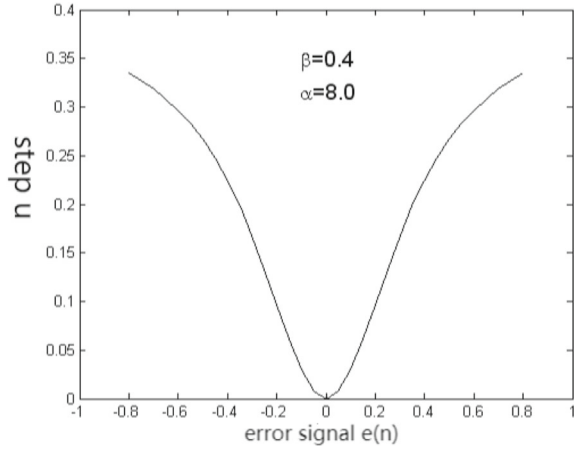
where the limit of the step length $\mu(n)$ and the shape of the step-length curve are determined by β . The function curve of step size $\mu(n)$ and error $e(n)$ is shown in Fig. 8.11. The limit of the step size is determined by β ; hence, the value of step size β must meet the convergence condition of the LMS algorithm.

f. VS-LMS-E-3 algorithm

Kwong et al. [14] proposed a new method to adjust the step size of the adaptive-filtering algorithm. This method generates a new step size using the weighted summation of the historical step-size values and the square of the current instantaneous error difference. The step-size adjustment formula is as follows:

$$\mu'(n+1) = \alpha\mu(n) + \gamma e^2(n) \quad (8.53)$$

Fig. 8.11 Function curve with step length $\mu(n)$ and error $e(n)$



where $0 < \alpha < 1$, $\gamma > 0$, and

$$\mu(n+1) = \begin{cases} \mu_{\max} & \mu'(n+1) > \mu_{\max} \\ \mu_{\min} & \mu'(n+1) < \mu_{\min} \\ \mu'(n+1) & \text{else} \end{cases} \quad (8.54)$$

where $0 < \mu_{\min} < \mu_{\max}$. The initial step value μ_0 is usually the maximum value μ_{\max} . We set $\mu_{\max} = 0.1$, $\mu_{\min} = 10^{-5}$, $\alpha = 0.97$, and $\gamma = 4.8 \times 10^{-4}$.

(2) Step size is related to the estimated value of the cross-correlation function of the error signal and the filter input vector

a. VS-LMS-EX-1 algorithm

To overcome the shortcomings of the SVS-LMS algorithm, Ref. [15] proposed a new variable-step-size algorithm. The step-size iteration formula of the new algorithm is as follows:

$$\mu(n) = \beta(1 - \exp(-\alpha\|e(n)\mathbf{x}(n)\|^2)) \quad (8.55)$$

In the initial stage, when the convergence speed of the algorithm $\|e(n)\mathbf{x}(n)\|$ is greatest, $\mu(n) \approx \beta$. As the algorithm approaches the steady state, $\|e(n)\mathbf{x}(n)\|$ decreases and $\mu(n)$ decreases; when entering the steady state, $\|e(n)\mathbf{x}(n)\|$ is very small, and the steady-state value of mean squared error $\mu(n)$ is also very small [15].

(b) VS-LMS-EX-2 algorithm

After analyzing the VS-LMS-E-3 algorithm proposed in Refs. [14], [16] points out that the VS-LMS-E-3 algorithm is still large in the late convergence stage, owing to the existence of input noise. $\mu(n)$ will make the weight coefficient fluctuate around the optimal value, resulting in a large imbalance. When the algorithm enters the steady

state, $e(n)$ is already very small. When the system parameters change suddenly, $\mu(n)$ is still very small and cannot well track the changes of the system [16].

To overcome the above shortcomings, $e(n)x(n)$ is introduced into the variable-step-size adjustment. The new $\mu(n)$ for iterative formula $\left(\sum_{n=1}^M e(n)x(n)/M\right)^2$ is as follows:

$$\mu'(n+1) = \alpha\mu(n) + \gamma \left(\sum_{n=1}^M e(n)x(n)/M \right)^2 \quad (8.56)$$

$$\mu(n+1) = \begin{cases} \mu_{\max} & \mu'(n+1) > \mu_{\max} \\ \mu_{\min} & \mu'(n+1) < \mu_{\min} \\ \mu'(n+1) & \text{else} \end{cases} \quad (8.57)$$

Because the step size in this algorithm is adjusted by the input signal $x(n)$ and output error $e(n)$, the algorithm has better tracking performance when the system parameters change.

(3) Step size is related to the autocorrelation estimation of the error signal

a. MDVS-LMS algorithm

The algorithm proposed in Ref. [13] has a simple operation, small step size, little change in the steady-state stage, and can achieve good adaptive-equalization results. However, this algorithm is vulnerable to the interference of uncorrelated noise at the signal input, which affects its stability. In view of the DVS-LMS algorithm, the MDVS-LMS-1 algorithm proposed in Refs. [29, 30] uses error signal $e(n)$ and $e(n-1)$ to adjust the step size. The step size of the MDVS-LMS-1 algorithm is adjusted as follows:

$$\mu(n) = \beta \left[1 - \frac{1}{\alpha \times |e(n) \times e(n-1)| + 1} \right] \quad (8.58)$$

The MDVS-LMS-1 algorithm uses error signal $e(n)$ plus $e(n-1)$ to adjust the step size, so that the algorithm convergence is not disturbed by uncorrelated noise at the signal input. However, in practical applications, the error signal $e(n)$ in the adaptive process may be irrelevant in the convergence stage, and the MDVS-LMS-1 algorithm may reduce the step size of the algorithm to the minimum before it converges [19]. The MDVS-LMS-1 algorithm is modified to obtain the step-adjustment formula of the MDVS-LMS-2 algorithm, as follows:

$$\mu(n) = \beta \left[1 - \frac{1}{\alpha \times |e(n) \times [e(n-1) + e(n)]| + 1} \right] \quad (8.59)$$

b. VS-LMS-EE-1 algorithm

Reference [17] proposed a new variable-step-size LMS algorithm, which uses the time mean estimation of the error signal to control the step-size update to ensure a faster dynamic-response speed. When updating the step size, the interference of uncorrelated noise signal $v(n)$ is greatly reduced, so that even with a large amount of noise interference, the system still has a small steady-state value for the mean squared error [17].

The step-size iteration formula of the VS-LMS-EE-1 algorithm is as follows:

$$p(n) = \beta p(n-1) + (1-\beta)e(n) \quad (8.60)$$

$$\mu(n+1) = \alpha \mu(n) + \gamma p(n) \quad (8.61)$$

$$\mu(n+1) = \begin{cases} \mu_{max} & \mu(n+1) > \mu_{max} \\ \mu_{min} & \mu(n+1) < \mu_{min} \\ \mu(n+1) & \text{else} \end{cases} \quad (8.62)$$

where $p(n)$ is called the time mean estimation of error signal $e(n)$, and $0 < \beta < 1$. $p(n)$ is called the forgetting factor, which is used to adjust the width of the time window to control the influence of past error signals on the current state. The earlier the data are, the greater the forgetting degree is, and the smaller its influence on the current state. Refer to Ref. [14] for a selection of parameters α and γ .

c. VS-LMS-EE-2 algorithm

After analyzing the VS-LMS-E-1 algorithm proposed in Refs. [10], [18] proposed an improved algorithm, which does not directly adjust the step size with the square of the signal error $|e(n)|^2$. Rather, it uses the correlation value of the error $|e(n)e(n-1)|$ to adjust the step size, which reduces the noise sensitivity of the variable-step-size LMS algorithm, has good anti-interference performance, and significantly improves its convergence and the steady-state value of the mean squared error [18].

The step-size iteration formula of the VS-LMS-E-1 algorithm is as follows:

$$\mu(n) = \beta(1 - \exp(-\alpha|e(n)e(n-1)|)) \quad (8.63)$$

d. VS-LMS-EE-3 algorithm

Observing that the VS-LMS-E-3 algorithm proposed in Ref. [14] cannot overcome the interference of uncorrelated noise at the input, Ref. [19] introduces the correlation value of error $e(n)e(n-1)$ to adjust the step size, so that the algorithm is not disturbed by uncorrelated noise. This algorithm is also introduced in Refs. [20, 21]. The step-size iteration formula of this algorithm is as follows:

$$p(n) = \beta p(n-1) + (1-\beta)e(n)e(n-1) \quad (8.64)$$

$$\mu(n+1) = \alpha\mu(n) + \gamma p^2(n) \quad (8.65)$$

The parameters in [19] are $\alpha = 0.97$, $\beta = 0.99$, and $\gamma = 1 \times 10^{-5}$ [19–21].

e. VS-LMS-EE-4 algorithm

The basic idea of the VS-LMS-EE-3 algorithm introduced above is that when the algorithm approaches the optimal value, not only does the adaptive error become very small; however, the correlation $e(n)$ between each sampling point is also very small. Therefore, the autocorrelation estimation is connected with the change of step factor $e(n)$; thus, in the initial adaptation stage, the autocorrelation estimation error $p^2(n)$ is large, resulting in a large step $\mu(n)$.

When the weight vector approaches the optimal value, the adaptive-estimation error $p^2(n)$ is close to zero, and the step size is small. However, in the high-order adaptive filter, the error $e(n)$ is not only uncorrelated when it is close to the optimal value, but also its correlation may be very small in the convergence process. This will lead to the step size of the adaptive algorithm quickly becoming smaller in the convergence process. A smaller step size increases the number of iterations and slows down the convergence speed.

Because the mean squared error is typically used to measure the convergence of the LMS algorithm, Ref. [22] adds the autocorrelation value $e(n)$ and the $e(n-1)$ instantaneous square error to make the estimate. The step iterative formula is as follows:

$$p(n) = \beta p(n-1) + (1-\beta)[e(n)e(n-1) + e^2(n)] \quad (8.66)$$

$$\mu(n+1) = \alpha\mu(n) + \gamma p^2(n) \quad (8.67)$$

where the selection of parameters α , β , and γ is the same as in the VS-LMS-EE-3 algorithm. Reference [23] also introduces the VS-LMS-EE-4 algorithm and applies it to a jammer adaptive-transceiver isolation system [22, 23].

f. VS-LMS-N-EE-1 algorithm

The normalized LMS (NLMS) algorithm can increase the dynamic-input range of the algorithm; however, the algorithm is very sensitive to noise. Reference [24] introduces the autocorrelation estimation of summing errors $e(n)$ and $e(n-1)$ in NLMS, which not only expedites the convergence speed and can track quickly in a non-stationary state, but also eliminates the interference of uncorrelated noise, and can adapt to a wide range of dynamic inputs [24]. The iterative formula of this algorithm is as follows:

$$e(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n) \quad (8.68)$$

$$p(n) = \beta p(n-1) + (1-\beta)e(n)e(n-1) \quad (8.69)$$

$$\mu(n+1) = \alpha\mu(n) + \gamma p^2(n) \quad (8.70)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu(n+1) \frac{1}{\|\mathbf{x}(n)\|^2} e(n) \mathbf{x}(n) \quad (8.71)$$

g. VS-LMS-N-EE-2 algorithm

Reference [25] proposed a new variable-step-size LMS algorithm that determines the next iteration step size by introducing correction coefficient ρ and forgetting factor λ , and using the current and past error information. In this manner, the known information is fully utilized to update the iterative step size from the perspective of nonlinear forward prediction, to adjust the filter-weight vector coefficient and make it quickly converge to the optimal value \mathbf{w}_{opt} .

The parameter adjustment of this algorithm is simple and the algorithm complexity increases only slightly, but the performance is greatly improved. References [26, 27] also proposed this algorithm. Its step-size iteration formula is as follows:

$$\mu(n) = \frac{1}{\frac{\rho}{|\lambda_n|} + \mathbf{x}^T(n)\mathbf{x}(n)} = \frac{1}{\frac{\rho}{|\lambda_n|} + \mathbf{R}(n)} \quad (8.72)$$

where $\rho > 0$ is the correction coefficient, which is equivalent to the role of γ in the NLMS algorithm.

$$\lambda_n = \sum_{i=0}^{N-1} \lambda(i)e(n-i) = \sum_{i=0}^{N-1} \exp(-i)e(n-i) \quad (8.73)$$

where λ_n is the error coefficient, in which $\lambda(i)$ is obtained by nonlinear weighting the past errors $e(n), e(n-1), \dots, e(n-N+1)$. The more past error information there is, the smaller the impact on the current error coefficient. $|\lambda_n|$ represents the deviation from the expected value after statistically weighting the current error. When $|\lambda_n|$ decreases, $\mu(n)$ decreases, allowing it to approach and slowly track the optimal value of the filter-weight vector. On the other hand, when $|\lambda_n|$ increases, $\mu(n)$ increases, to quickly approach the optimal value. Each iteration of this algorithm needs $\mathbf{R}(n)$ calculations:

$$\mathbf{R}(n) = \mathbf{x}^T(n)\mathbf{x}(n) = x^2(n) + x^2(n-1) + \dots + x^2(n-N+1) \quad (8.74)$$

The number of calculations is directly proportional to the number of filter taps. In a practical implementation, we can make full use of the previous calculation value $\mathbf{R}(n-1)$ to calculate $\mathbf{R}(n)$:

$$\mathbf{R}(n) = \mathbf{R}(n-1) + x^2(n) - x^2(n-N) \quad (8.75)$$

This method of reducing the calculation amount is also introduced in Ref. [28], which calls it a sliding time window.

When calculation λ_n , $e(n)$, $e(n-1)$, \dots , $e(n-N+1)$ are known and do not need to be recalculated, and $\lambda(i) = \exp(-i)$ are constants, which can be calculated in advance during the algorithm initialization. In a practical implementation, the above methods can greatly reduce the amount of calculation and further improve the performance [25–28].

h. VS-LMS-N-EE-3 algorithm

Reference [28] proposed a variable-step-size LMS algorithm that calculated the step size according to the autocorrelation of the gradient difference of the filter coefficients. The iterative formula of the algorithm is as follows:

$$q(n) = \left| \frac{e(n)e(n-1)\mathbf{x}^T(n-1)\mathbf{x}(n)}{\gamma + \|\mathbf{x}(n-1)\|^2\|\mathbf{x}(n)\|^2} - p(n-1) \right| \quad (8.76)$$

$$p(n) = q(n)q(n-1) \quad (8.77)$$

$$\mu(n) = \alpha\mu(n-1) + \beta p(n) \quad (8.78)$$

$$\mu = \begin{cases} \mu_{\max} & \mu > \mu_{\max} \\ \mu_{\min} & \mu < \mu_{\min} \\ \mu & \text{else} \end{cases} \quad (8.79)$$

where $0 < \mu_{\min} < \mu_{\max}$, and the initial step size μ_{\max} is taken as a small positive value γ to ensure that the denominator is not zero. Parameter α plus β in Eq. (8.78) is a value within the range (0, 1), which is an exponential window, representing the forgetting coefficient and reflecting the reliability of the $p(n)$. The parameter values in this study are $\alpha = 0.98$, $\beta = 0.98$, $\mu_{\max} = 0.2$, and $\mu_{\min} = 1 \times 10^{-5}$ [28].

4. Variable-order LMS algorithms

Various variable-step-size LMS algorithms were introduced above. Reference [31] explores another method to change the filter order. Here, the order of the adaptive filter N is a variable $N(n)$, $\mathbf{w}(n)$ and $\mathbf{x}(n)$ are the dimensions of $N(n)$ and change accordingly. If the available orders L are arranged from small to large, N_1, N_2, \dots, N_L , it is necessary to establish a variable-order criterion, $\xi(n) = f(e(n))$. Then, the relationship expression between the error and order can be established, such as the piecewise function with the order, as error $N(n+1) = s(\xi(n))$.

An error-evaluation function is first established. Then, the value interval is divided into subintervals corresponding to the order, and a piecewise function is established. These two functions are reasonably selected so that the smaller order is taken when the error is large and the larger order is taken when the error is small, and the optional order is fully utilized. When the order changes, the dimension can be changed by simply changing the number of recent sampling values $\mathbf{x}(n)$, which is expressed by

a function. The transition function $\mathbf{x}(n+1) = h(\mathbf{x}(n))$ can be used with different dimensions:

$$\mathbf{w}^*(n) = g(\mathbf{w}(n), N(n+1)) \quad (8.80)$$

This representation is called the weight-vector inheritance function.

When changing from a small order to a large order, such as from N_i ($1 \leq i < j \leq L$) to N_j , assuming that the original weight vector is $\mathbf{w}(n) = [w_i, \dots, w_2, w_1]$, we move the original component $\mathbf{w}(n)$ backward by $j-i$ bits, and fill the new previous component $j-i$ with zeroes; then, a new weight vector N_j is obtained to replace the iterative operation of the continuation algorithm $\mathbf{w}^*(n) = [0, \dots, 0, w_i, \dots, w_2, w_1]$. When changing from a large order to a small order, if the order changes, the first component can be rounded off and the next component can be retained.

The error-evaluation function is selected as $\xi(n) = \rho \exp(-\gamma e^2(n))$, and the shape of the function depends on parameters ρ and γ . They determine the amplitude and steepness of the curve.

Figure 8.12a and b respectively show the error evaluation function curves when selecting different values of ρ and γ

The piecewise function can be selected as the curve shown in Fig. 8.13 ($\rho = 5$ and the optional order is 7, 9, 11, 13, and 15). Only the steepness of the evaluation function γ needs to be changed, so that the appropriate minimum order is adopted within a certain large error range. The appropriate maximum order is adopted when the error is close to zero, and there is a moderate intermediate-order changing process.

Changing the step size while changing the order is accomplished by the variable-order adaptive filter of the variable-step-size LMS algorithm. The iterative formula of the algorithm is as follows:

$$e(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n) \quad (8.81)$$

$$\mu(n) = \beta(1 - \exp(-\alpha e^2(n))) \quad (8.82)$$

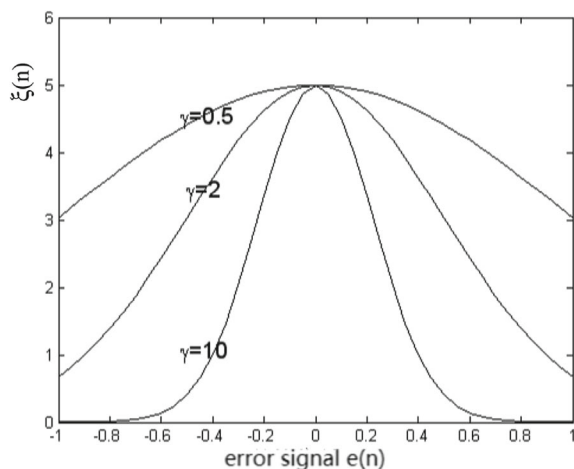
$$\xi(n) = \rho \exp(-\gamma e^2(n)) \quad (8.83)$$

$$N(n+1) = s(\xi(n)) \quad (8.84)$$

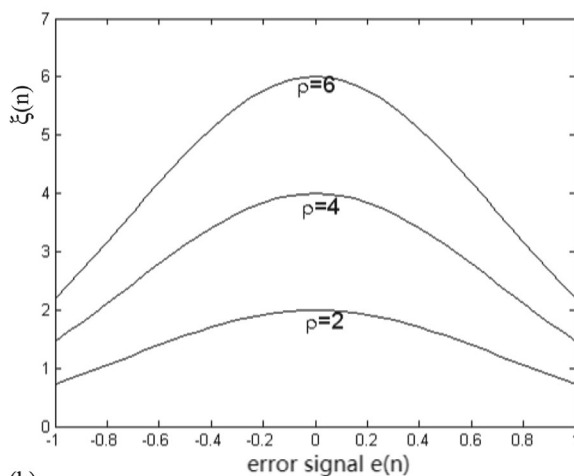
$$\mathbf{w}(n+1) = g(\mathbf{w}(n), N(n+1)) - 2\mu(n)e(n)h(\mathbf{x}(n), N(n+1)) \quad (8.85)$$

The form of this algorithm is very complex; however, the inheritance function and the piecewise function are simple rules that are easy to implement. The evaluation function does not need accurate values and is easy to implement. The purpose of the variable step size and order is to select a lower order and large step size at the initial learning stage to expedite the convergence speed, and select a higher order and small step size when the algorithm is close to convergence, to reduce the imbalance [31].

Fig. 8.12 Error-evaluation function during changes of γ and ρ **a** Error-evaluation function with different values of γ , where $\rho = 5$ **b** Error-evaluation function with different values of ρ , where $\gamma = 1$



(a)



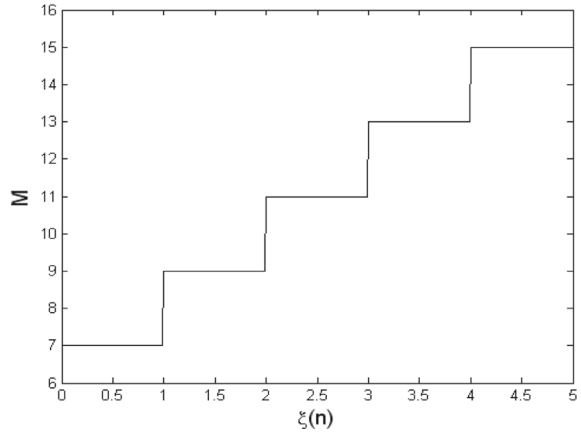
(b)

8.3 Adaptive Equalizers

8.3.1 Adaptive Linear Equalizers

If the channel-transmission characteristics of a communication system are not ideal, the transmission signal will be distorted, resulting in inter symbol interference (ISI). The circuit that corrects this undesirable characteristic is called an equalizer. An adaptive equalizer can automatically adjust the parameters, according to the changes in the channel-transmission characteristics, to achieve the best equalization possible

Fig. 8.13 Piecewise function $N(n+1) = s(\xi(n))$



[2]. Adaptive-equalization technology is highly significant for atmospheric laser channels.

The information to be transmitted $I(n)$ is encoded and output after modulation as $x(n)$, to the transmitting optical antenna, channel, receiving optical antenna, and demodulator. The system model and linear equalizer are shown in Fig. 8.14.

As can be seen from Fig. 8.14, the input of the linear equalizer is

$$x(n) = \sum_{i=0}^{N-1} h_i I(n-i) + v(n) \quad (8.86)$$

The output of the equalizer $\hat{I}(n)$ is an estimate of $I(n)$:

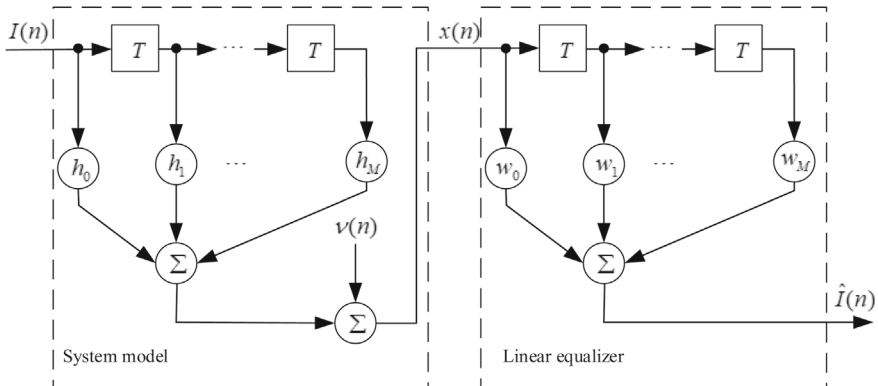


Fig. 8.14 System model and linear equalizer

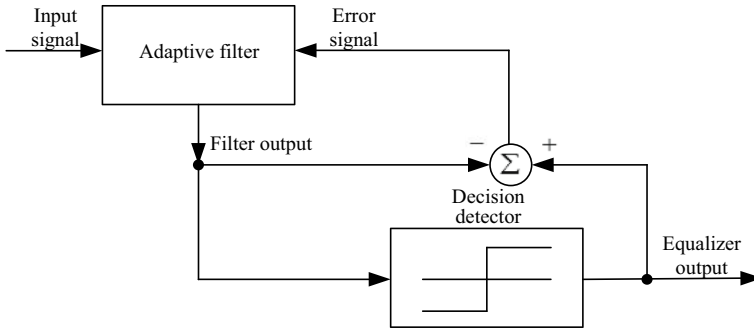


Fig. 8.15 Adaptive equalizer with a decision detector

$$\hat{I}(n) = \sum_{i=0}^{N-1} w_i x(n-i) \quad (8.87)$$

8.3.2 Decision-Guided Adaptive Equalizer

The LMS algorithm must know the reference signal $d(n)$. To obtain it, in the equalizer-training stage, the sender periodically sends a training sequence known by the receiver, and the receiver uses the training sequence to update the equalizer weight-vector coefficient to minimize the mean squared error (MSE) of the equalized error signal. In the working stage of the equalizer, decision detector $y(n)$ is used to directly generate the reference signal $d(n)$ from the filter output. At this time, error signal $e(n)$ is the decision error. This method is called the decision-guidance method [7].

Figure 8.15 shows a decision detector for digital information suitable for two-level quantization. Figure 8.16 shows a more detailed block diagram of the LMS adaptive equalizer.

In a practical application, the equalizer first enters the training stage to optimize its weight-vector coefficient as much as possible. Then, the equalizer enters the working stage; and the information structure of the transmission sequence is shown in Fig. 8.17. The longer the training sequence, the smaller the bit error rate; however, it also leads to a decline in spectrum utilization.

8.3.3 Adaptive Fractional-Interval Equalizer

The tap interval of the adaptive linear equalizer is equal to the signal symbol width T . The actual channel response is generally unknown, and the receiving filter can only match the transmitted pulse. At this point, the performance of the equalizer is

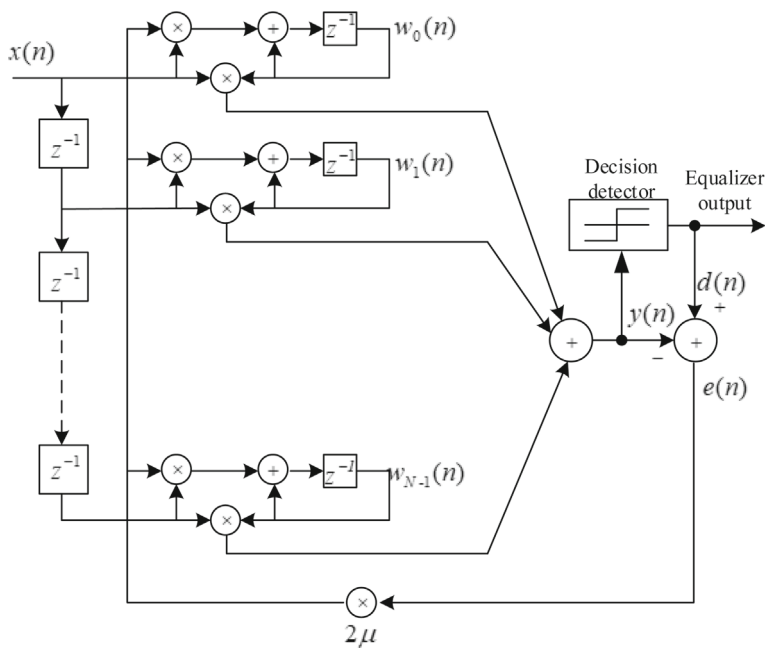


Fig. 8.16 Adaptive LMS decision-guided equalizer

Fig. 8.17 Information structure of a transmission sequence



very sensitive to the sampling time. A fractionally spaced equalizer (FSE) can solve this problem.

The transverse-filter tap interval of the FSE is MT/N , where $N > M$. The most commonly used FSE uses a tap spacing of $T/2$. The schematic block diagram of an FSE with a tap interval of $T/2$ is shown in Fig. 8.18. In the figure, the input data rate of the equalizer is T , the input signal is sampled according to the rate, the tap interval of the adaptive filter is $T/2$, and the adaptive equalizer samples at the rate of $1/T$ to recover the input data rate. The reference signal for adaptive filtering is generated by the decision detector. The sampling rate of the error signal is also $1/T$ [7].

The FSE can improve the timing sensitivity and filtering performance of the equalizer at the cost of increasing the number of equalizer taps. For tap cases $T/2$, the number of taps is doubled.

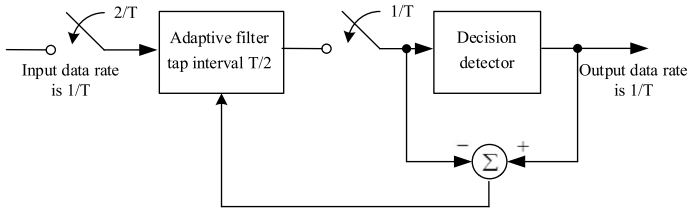


Fig. 8.18 Fractional-interval equalizer

8.3.4 Blind Equalizers

(1) Definition of blind equalization

Figure 8.19 shows the model of a blind-equalization system. Blind equalization does not rely on a training sequence; rather, it only uses the prior information of the received signal itself, such as the statistical characteristics, modulation mode, and amplitude of the signal, phase-variation range, etc., to select an appropriate cost function and error-control quantity to adjust the equalizer weight-vector coefficient, so that the equalizer output is closest to the transmitted signal [2].

(2) Bussgang blind equalizer

The Bussgang algorithm uses a zero-memory nonlinear function $y(n)$ output by the filter, as the estimation of the reference signal $g[y(n)]$.

If a random process $y(n)$ satisfies

$$E\{y(n)y(n-k)\} = E\{y(n)g[y(n-k)]\} \quad (8.88)$$

$y(n)$ is called a Bussgang process. For blind-equalization processing, the filter output $y(n)$ approximately satisfies Eq. (8.87). Therefore, this blind-equalization algorithm is called the Bussgang algorithm. The block diagram of a Bussgang blind equalizer using the LMS algorithm is shown in Fig. 8.20 [2].

(3) Constant-modulus algorithm

The constant-modulus algorithm (CMA) is a type of Bussgang blind-equalization algorithm. The performance function defined by the constant-modulus algorithm is

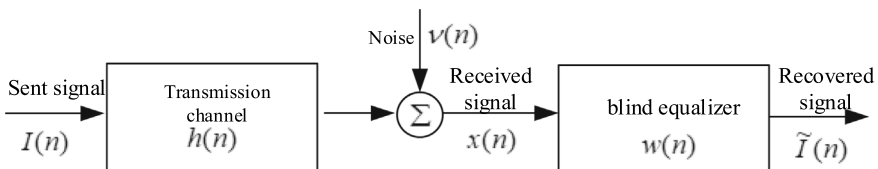


Fig. 8.19 Blind-equalization system model

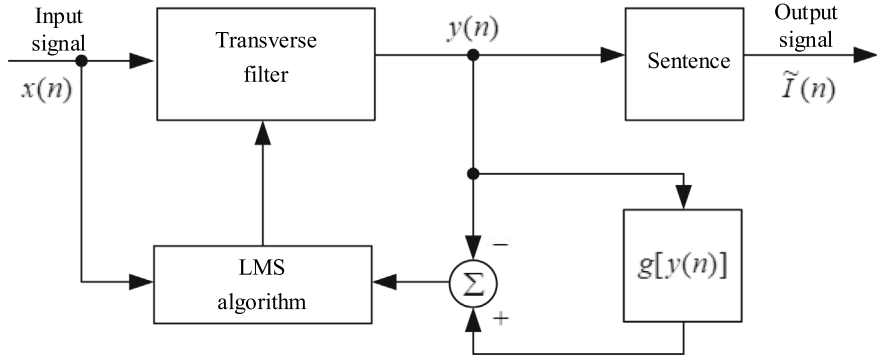


Fig. 8.20 Busgang blind equalizer using the LMS algorithm

$$\xi = E\{|y(n)|^p - 1\}^q \quad (8.89)$$

where p and q are positive integers, which are taken as 1 or 2 in practice, and recorded as CMA_{p-q} accordingly. The iterative formula of the constant-modulus algorithm based on LMS is

$$w(n+1) = w(n) + \mu x(n)e(n) \quad (8.90)$$

where

$$CMA_{1-1} : e(n) = \frac{y(n)}{|y(n)|} \text{sgn}(|y(n)| - 1) \quad (8.91)$$

$$CMA_{2-1} : e(n) = 2y(n)\text{sgn}(|y(n)|^2 - 1) \quad (8.92)$$

$$CMA_{1-2} : e(n) = 2\frac{y(n)}{y(n)}\text{sgn}(|y(n)| - 1) \quad (8.93)$$

$$CMA_{2-2} : e(n) = 4y(n)\text{sgn}(|y(n)|^4 - 1) \quad (8.94)$$

Among the above four formulas, CMA_{1-2} and CMA_{2-2} are the most commonly used [32–34]. Similar to the LMS algorithm, in the Busgang blind-equalization algorithm, the step size plays a very important role in the algorithm-convergence process. If a large step size is adopted, the amplitude of the tap coefficient will be large each time. The convergence performance is reflected in the fast convergence and tracking speeds of the algorithm. When the tap coefficient of the equalizer is close to the optimal value, it will jitter back and forth in a large range near the optimal value without converging further; thus, there will be a large steady-state value of the mean squared error.

On the other hand, with a small step length, the amplitude of each tap-coefficient adjustment is small, and the convergence and tracking speeds of the algorithm are slow. However, when the tap coefficient of the equalizer is close to the optimal value, the tap coefficient will fluctuate in a small range near the optimal value; thus, the steady-state value of the mean squared error is small.

In the CMA algorithm, the iteration step factor is a fixed constant, which makes a contradiction between the convergence speed, tracking speed, and steady-state value of the mean squared error. To address this contradiction, we can also adopt the variable-step-size method; that is, increase the step size at the initial algorithm-convergence stage to expedite the convergence speed, and reduce the step size when approaching convergence to improve the convergence accuracy. The iterative formula of the weight-vector coefficient of a variable-step-size calculation method is

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \mu(n)\mathbf{x}(n)e(n) \quad (8.95)$$

$$\mu(n+1) = \mu(n) + \beta[\text{MSE}(n+1) - \text{MSE}(n)] \quad (8.96)$$

(4) Dual-mode blind-equalization algorithm

The dual-mode blind equalizer using the LMS algorithm is shown in Fig. 8.21, where $x(n)$ is the input signal of the blind equalizer after the transmission signal; $I(n)$ is transmitted through the channel; and $y(n)$ is the output signal of the transverse filter inside the blind equalizer. After signal $y(n)$ is judged, the output signal $\tilde{I}(n)$ of the blind equalizer is obtained; that is, $I(n)$, the estimated value of the transmitted signal. $g[y(n)]$ is a memoryless nonlinear function and $e(n)$ is the error signal [36–39]. The iterative process of the blind equalizer based on the LMS algorithm is as follows:

$$y(n) = \mathbf{w}^T(n)\mathbf{x}(n) \quad (8.97)$$

$$e(n) = R - |y(n)| \quad (8.98)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{x}(n)e(n) \quad (8.99)$$

where $\mathbf{w}(n)$ is the filter weight-vector coefficient, μ is the step factor, and R is a constant:

$$R = \frac{E[|x(n)|]^4}{E[|x(n)|]^2} \quad (8.100)$$

The error signal when the blind equalizer switches to the working state of the decision-guidance equalizer is

$$e(n) = \tilde{I}(n) - y(n) \quad (8.101)$$

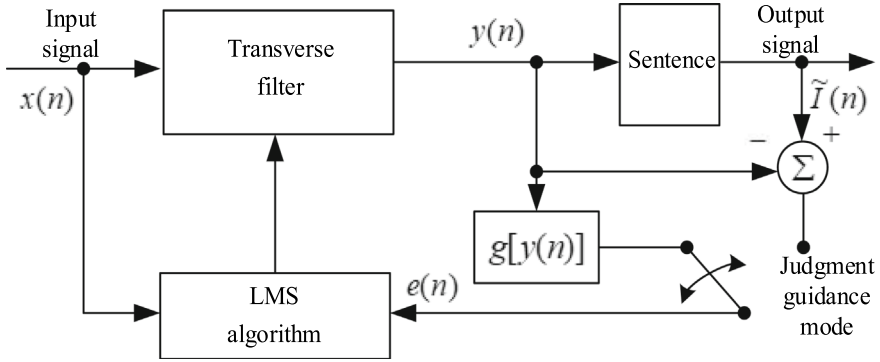
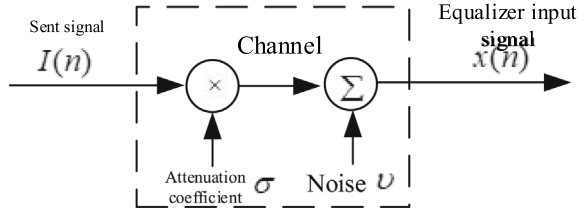


Fig. 8.21 Dual-mode blind equalizer using the LMS algorithm

Fig. 8.22 Atmospheric laser channel model



As shown in Fig. 8.21, in the initial stage of the equalizer, the error signal $e(n)$ is generated by $g[y(n)]$, and the filter weight-vector coefficient is updated using Eqs. (8.97), (8.98), and (8.99). Subsequently, the filter enters the working stage of the decision-guidance equalizer, updates the weight-vector coefficients using Eqs. (8.97), (8.99), and (8.101), and performs decision detection on the output $y(n)$ of the filter. It uses the decision result as the reference signal $\tilde{I}(n)$, and the error signal $e(n)$ is the decision error.

8.4 Adaptive-Equalization Simulations of Wireless Laser Communications

8.4.1 Adaptive-Equalization Simulations

(1) Simulation environment

Through a large number of actual measurements, the power data at the receiving end were detected when the transmission power was constant at 10 MW in fog, overcast sky, and light, moderate, and heavy rain [40].

The received power value was normalized to obtain the channel attenuation coefficient σ to the transmitted signal. The attenuation coefficient σ was used to modulate

the transmission signal $I(n)$, including additive Gaussian white noise with a mean value of 0 and variance of 0.01, to obtain the input signal $x(n)$ of the filter.

The schematic diagram of an atmospheric laser channel model is shown in Fig. 8.22. To compare the performance of various algorithms, it is necessary to reflect the generality of the simulation experiment. The channel characteristics are given by vector ISI_1 [41]:

$$ISI_1 = (0.05, -0.063, 0.088, -0.126, -0.25, 0.9047, 0.25, 0, 0.126, 0.038)$$

To observe the tracking performance of the adaptive-filtering algorithm, the channel changes after a certain number of iterations; its channel-characteristic vector is $ISI_2 = (-0.25, 0.9047, 0.038)$.

After the transmission signal is processed by the constant parameter channel, including additive Gaussian white noise with a mean value of 0 and variance of 0.01, it is used as the input signal of the adaptive filter. The block diagram of the adaptive equalization experiment is shown in Fig. 8.23. In the experiment, considering the adaptive filter set, each LMS algorithm adopts the step parameters of the same adjustment mode, and uses the same initial tap-weight vector $\mathbf{w}(0) = (0, 0, 0, 0, 0, 1, 0, 0, 0, 0)$. Using the relationship between the mean squared amplitude of the estimation error $e(n)$ and the number of iterations, the average learning curve of the adaptive filter set can be drawn. The mean squared error curves were obtained by 500 independent simulations of each algorithm to obtain the adaptive filter set, and then the set was averaged [2].

(2). Simulation of the LMS algorithm

Research on the LMS algorithm mainly includes its convergence and the steady-state value of the mean squared error of the algorithm. The step size μ plays a key role in these two aspects.

The experiment is carried out according to the structure in Fig. 8.22; 500 independent simulations are carried out for each step μ , and then the set is averaged to

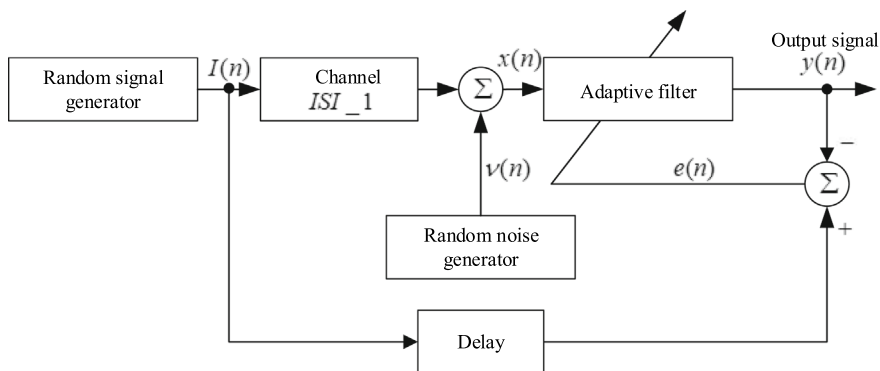
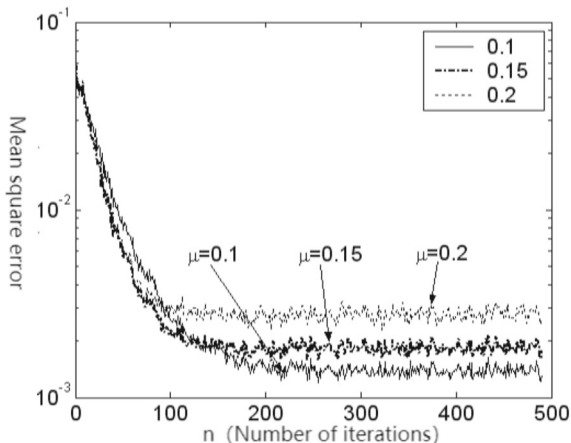


Fig. 8.23 Adaptive-equalization experimental block diagram #1

Fig. 8.24 Influence of the step size μ on the mean squared error of the LMS algorithm. μ equals 0.1, 0.15, and 0.2 [29]



obtain the mean squared error curve. The mean squared error (MSE) curve, when the step size μ is 0.1, 0.15, and 0.2, is shown in Fig. 8.24.

As shown in Fig. 8.24, when the step size μ is 0.2, the convergence speed of the LMS algorithm is rapid and the steady-state value of the mean squared error is large. When the step size μ is 0.1, the convergence speed of the LMS algorithm decreases; however, the steady-state value of the mean squared error also decreases. This reflects the contradiction between the convergence speed and the steady-state value of the mean squared error when the step size is fixed.

The filter length is also one of the factors affecting the convergence performance of the algorithm. The longer the filter, the greater the computational complexity. The influence of the filter length on the LMS algorithm is shown in Fig. 8.25. As can be seen from the figure, when the filter length is 10 and 11, the convergence speed is the same; however, when the length is 11, the steady-state value of the mean squared error is smaller.

(3). Simulation of the normalized LMS algorithm

The iterative formula of the weight-vector coefficient of the normalized LMS (NLMS) algorithm is as follows:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\mu}{\gamma + \|\mathbf{x}(n)\|^2} \mathbf{x}(n)e(n) \quad (8.102)$$

Five-hundred independent simulations were conducted of the normalized LMS and LMS algorithms, and then the set was averaged to obtain the mean squared error curve. The comparison is shown in Fig. 8.26, where the value of μ is 0.45 and the value of γ is 1×10^{-4} . The step size of the LMS algorithm is 0.1.

It can be seen from Fig. 8.26 that the normalized LMS algorithm converges faster than the LMS algorithm; however, the steady-state value of the mean squared error is relatively large.

Fig. 8.25 Influence of the filter length on the mean squared error of the LMS algorithm when $\mu = 0.1$. The filter length N is 9, 10, and 11 [29]

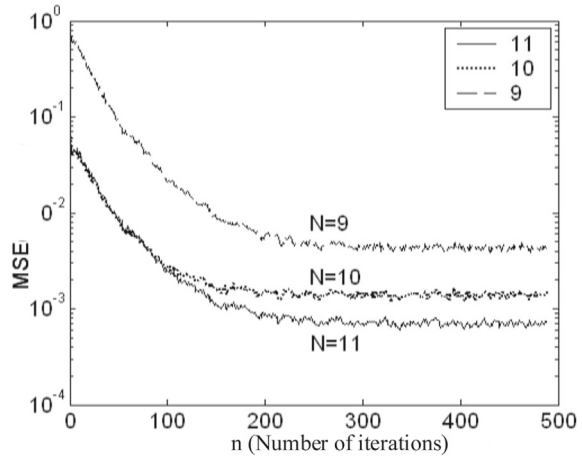
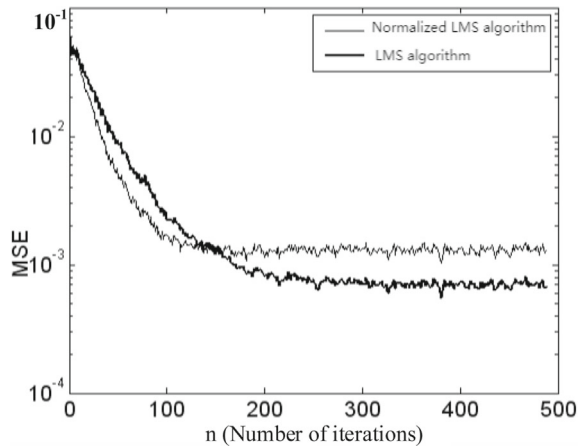


Fig. 8.26 Comparison between the normalized LMS and LMS algorithms [29]



(4) Simulation of simplified LMS algorithms

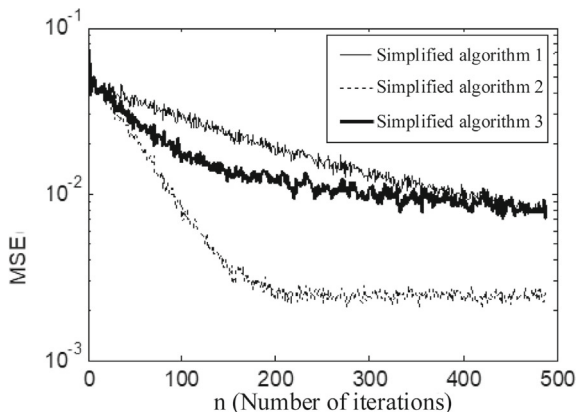
The iterative formulas of the weight-vector coefficient of the simplified LMS algorithm are as follows:

$$\text{Simplified algorithm 1 : } \mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu \text{sign}[\mathbf{x}(n)]e(n) \quad (8.103)$$

$$\text{Simplified algorithm 2 : } \mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu \mathbf{x}(n) \text{sign}[e(n)] \quad (8.104)$$

$$\text{Simplified algorithm 3 : } \mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu \text{sign}[\mathbf{x}(n)] \text{sign}[e(n)] \quad (8.105)$$

Fig. 8.27 Performance comparison of three simplified LMS algorithms [29]



Five-hundred independent simulations were conducted for each of the three algorithms, and then the set was averaged to obtain the mean squared error curve, as shown in Fig. 8.27, where the value of μ is 0.01.

As can be seen from Fig. 8.27, simplified algorithm 2 can converge quickly and the steady-state value of the mean squared error is small. The steady-state values of the mean squared error of simplified algorithms 1 and 3 are large.

(5) Simulation of variable-step-size LMS algorithms [42–45]

In the simulation and comparison of the variable-step-size LMS algorithms, the algorithm tracking characteristics when the channel characteristics change are considered, so the channel characteristics are first given by the vector. After a certain number of iterations, the channel characteristics change, and the vector is $ISI_2 = (-0.25, 0.9047, 0.038)$. After the transmission signal is processed by the constant-parameter channel, including additive Gaussian white noise with a mean value of 0 and variance of 0.01, it is used as the input signal of the adaptive filter. The block diagram of the adaptive-equalization experiment is shown in Fig. 8.28.

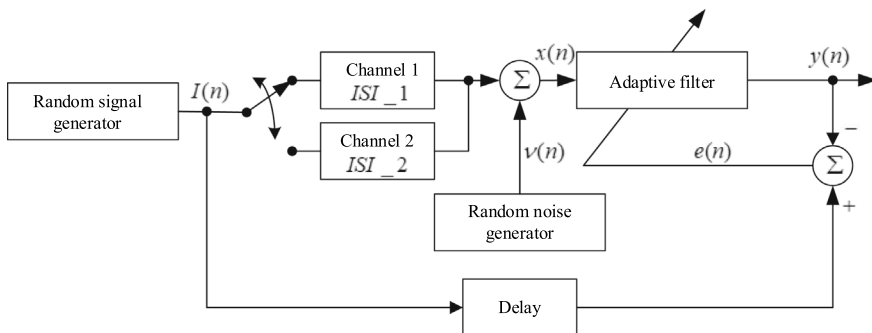
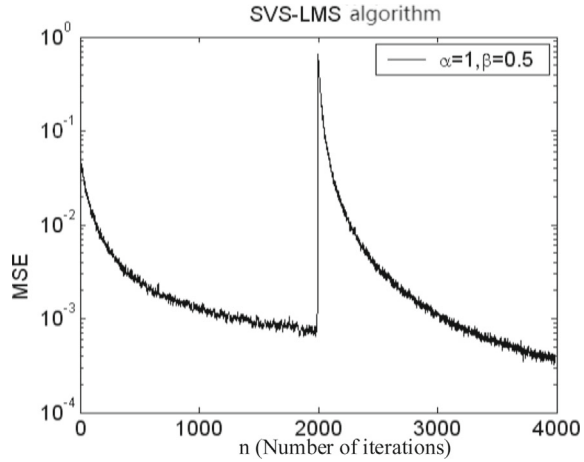


Fig. 8.28 Block diagram of adaptive-equalization experiment #2

Fig. 8.29 Mean squared error curve of the SVS-LMS algorithm [29]



a. **Algorithm for establishing the relationship between the step size and error signal**

i. **SVS-LMS algorithm**

$$\mu(n) = \beta \left(\frac{1}{1 + \exp(-\alpha|e(n)|)} - 0.5 \right) \quad (8.106)$$

The simulation is carried out according to the structure in Fig. 8.28. The channel characteristics change after 2000 points. Different experimental α and β affect the convergence speed, steady-state value of the mean squared error, and tracking speed. When parameter value α is 1 and β is 0.5, the performance of the algorithm is the best. Five-hundred independent simulations of the SVS-LMS algorithm were conducted, and the set was averaged to obtain the mean squared error curve, as shown in Fig. 8.29.

(ii) **VS-LMS-E-1 algorithm**

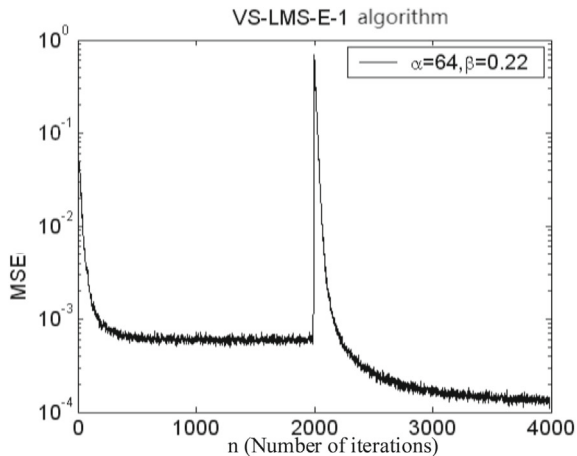
$$\mu(n) = \beta (1 - \exp(-\alpha|e(n)|^2)) \quad (8.107)$$

The simulation was carried out according to the structure in Fig. 8.28. The channel characteristics changed after 2000 points. Different experimental α and β affect the convergence speed, steady-state value of the mean squared error, and tracking speed. When α is 64 and β is 0.22, the performance of the algorithm is the best. Five-hundred independent simulations of the VS-LMS-E-1 algorithm were conducted, and the set was averaged to obtain the mean-squared-error curve, as shown in Fig. 8.30.

(iii) **DVS-LMS algorithm**

$$\mu(n) = \beta \left[1 - \frac{1}{\alpha \times e^2(n) + 1} \right] \quad (8.108)$$

Fig. 8.30 Mean squared error curve of the VS-LMS-E-1 algorithm [29]

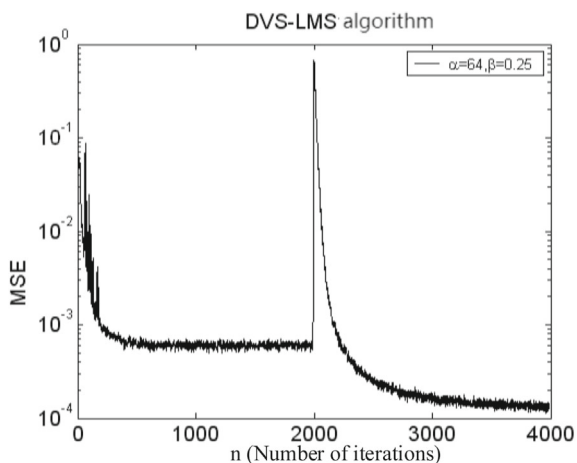


The simulation was carried out according to the structure in Fig. 8.28. The channel characteristics changed after 2000 points. Different experimental α and β affect the convergence speed, steady-state value of the mean squared error, and tracking speed. When α is 64 and β is 0.25, the performance of the algorithm is the best. Five-hundred independent simulations of the DVS-LMS algorithm were conducted, and the set was averaged to obtain the mean squared error curve, as shown in Fig. 8.31.

(iv) **VS-LMS-E-3 algorithm**

$$\mu(n+1) = \alpha\mu(n) + \gamma e^2(n) \quad (8.109)$$

Fig. 8.31 Mean squared error curve of the DVS-LMS algorithm [29]



$$\mu(n+1) = \begin{cases} \mu_{\max} & \mu'(n+1) > \mu_{\max} \\ \mu_{\min} & \mu'(n+1) < \mu_{\min} \\ \mu'(n+1) & \text{else} \end{cases} \quad (8.110)$$

where $\mu_{\max} = 0.1$ and $\mu_{\min} = 0.005$.

The simulation is carried out according to the structure in Fig. 8.28. The channel characteristics change after 2000 points. Different experimental α and β affect the convergence speed, steady-state value of the mean squared error, and tracking speed. When α is 0.98 and $\gamma = 0.9$, the performance of the algorithm is the best. Five-hundred independent simulations of the VS-LMS-E-3 algorithm were conducted, and then the set was averaged to obtain the mean squared error curve, as shown in Fig. 8.32.

A comparison of the above four algorithms with the best performance is shown in Fig. 8.33.

It can be seen from Fig. 8.33 that the four variable-step-size LMS algorithms converge well. Among them, the steady-state value of the mean squared error of the SVS-LMS algorithm is the largest, and the tracking speed is the slowest when the channel characteristics change. The steady-state values of the mean squared error of the other three algorithms are small and the tracking speed is fast, which reflects the performance improvement of these three algorithms over the SVS-LMS algorithm.

In practical applications, the SVS-LMS and VS-LMS-E-1 algorithms require an exponential operation, and the number of operations is relatively large. The DVS-LMS algorithm is unstable in the initial stage. The VS-LMS-E-3 algorithm has a small steady-state value for the mean squared error and simple operation, which can easily realize real-time signal processing.

The channel changes after 2000 points. We observe the number of bit errors of the VS-LMS-E-3 algorithm. The decision threshold is selected, and the number of error bits of this algorithm is shown in Fig. 8.33a and b.

Fig. 8.32 Mean squared error curve of the VS-LMS-E-3 algorithm [29]

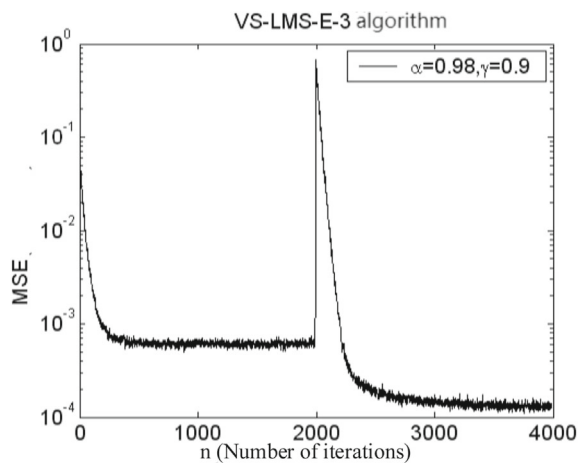
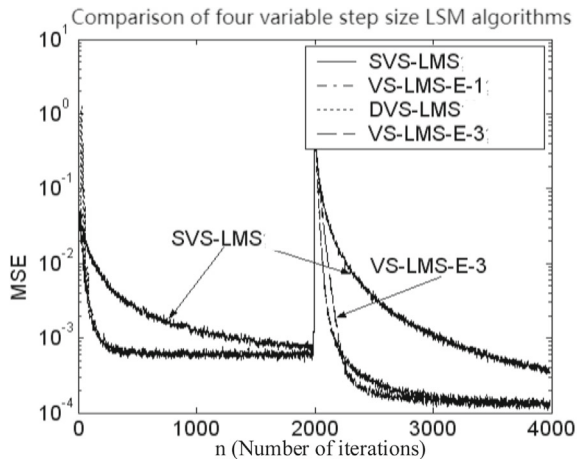


Fig. 8.33 Comparison of the four algorithms with optimal performance [29]



As can be seen from Fig. 8.34a and b, before the channel characteristics change, the VS-LMS-E-3 algorithm has a good equalization effect without bit errors. After the channel characteristics change, the filter enters the tracking stage, and the tap coefficient converges quickly. The convergence is completed within 40 iterations, and no new misjudgments occur, which reflects the good tracking performance of the algorithm.

(b) **Algorithm for establishing the relationship between the step size $e(n)$ and the estimated value $x(n)$ of the sum cross-correlation function.**

i. **VS-LMS-EX-1 algorithm**

$$\mu(n) = \beta(1 - \exp(-\alpha \|e(n)x(n)\|^2)) \quad (8.112)$$

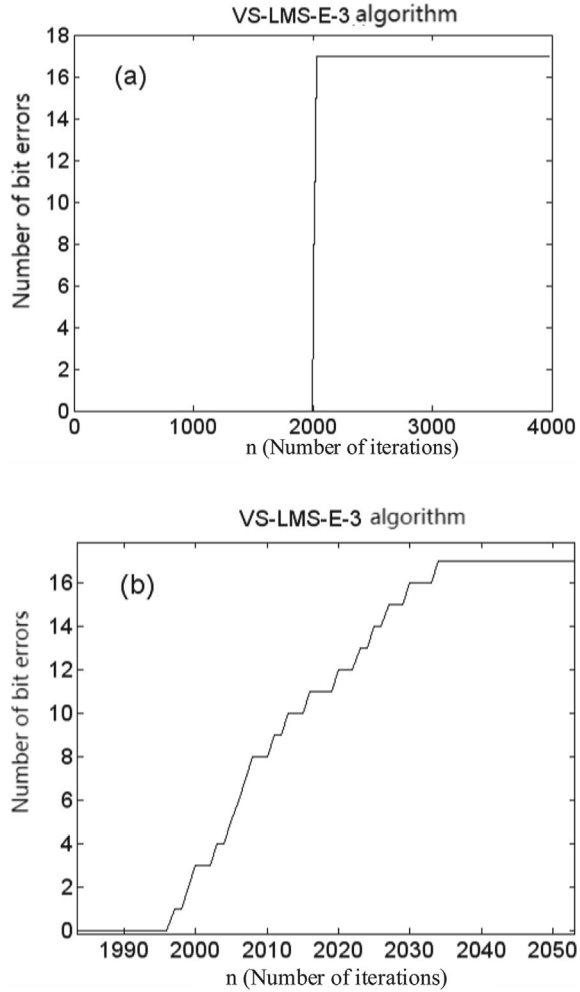
The simulation is carried out according to the structure in Fig. 8.28. The channel characteristics change after 2000 points. The performance of the algorithm is the best when different parameters have different effects on the convergence speed, steady-state value of the mean squared error, and tracking speed. We selected $\alpha = 32$ and $\beta = 0.25$. Five-hundred independent simulations of the VS-LMS-EX-1 algorithm were conducted, and the set was averaged to obtain the mean squared error curve, as shown in Fig. 8.35.

ii. **VS-LMS-EX-2 algorithm**

$$\mu'(n+1) = \alpha\mu(n) + \gamma \left(\sum_{n=1}^M e(n)x(n) / M \right)^2 \quad (8.113)$$

$$\mu(n+1) = \begin{cases} \mu_{max} & \mu'(n+1) > \mu_{max} \\ \mu_{min} & \mu'(n+1) < \mu_{min} \\ \mu'(n+1) & \text{else} \end{cases} \quad (8.114)$$

Fig. 8.34 Number of bit errors and local characteristics of the VS-LMS-E-3 algorithm
a Number of bit errors in the VS-LMS-E-3 algorithm [29]
b Partial enlarged view [29]

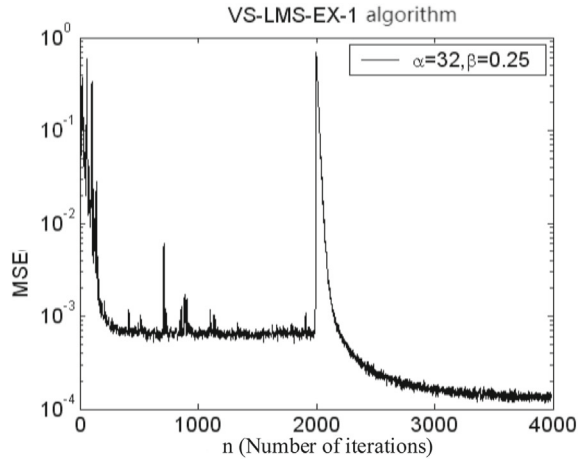


where $\mu_{max} = 0.1$ and $\mu_{min} = 0.005$.

The simulation is carried out according to the structure in Fig. 8.28. The channel characteristics change after 2000 points. The algorithm performance is the best when different parameters have different effects on the convergence speed, steady-state value of the mean squared error, and tracking speed. We selected $\alpha = 0.99$ and $\gamma = 1$. Five-hundred independent simulations of the VS-LMS-EX-2 algorithm were conducted, and the set was averaged to obtain the mean squared error curve, as shown in Fig. 8.36.

A comparison of the above two algorithms with the best performance is shown in Fig. 8.37. As can be seen from Fig. 8.36, the steady-state values of the mean squared error of the two algorithms are very close and converge well. However, in the initial stage, the mean squared error of the VS-LMS-EX-1 algorithm is large and

Fig. 8.35 Mean squared error curve of the VS-LMS-EX-1 algorithm [29]



the performance is unstable. When the channel characteristics change, the tracking speed of the VS-LMS-EX-1 algorithm is faster. From Eqs. (8.112) and (8.113), it can be seen that the computation of the VS-LMS-EX-2 algorithm is relatively small.

c. Step size is related to the autocorrelation estimation of the error signal

iii. MDVS-LMS-1 algorithm

$$\mu(n) = \beta \left[1 - \frac{1}{\alpha \times |e(n-1) \times e(n)| + 1} \right] \quad (8.115)$$

Fig. 8.36 Mean squared error curve of the VS-LMS-EX-2 algorithm [29]

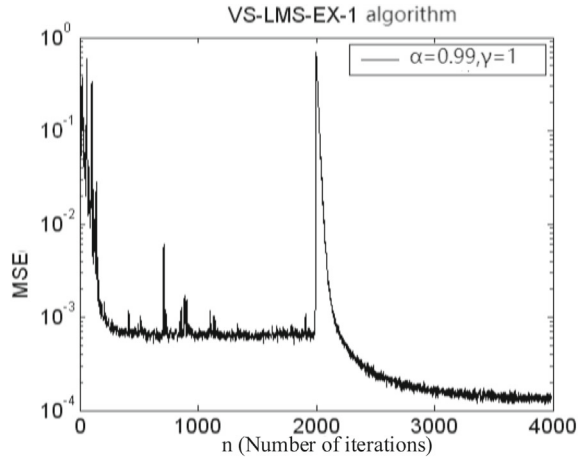
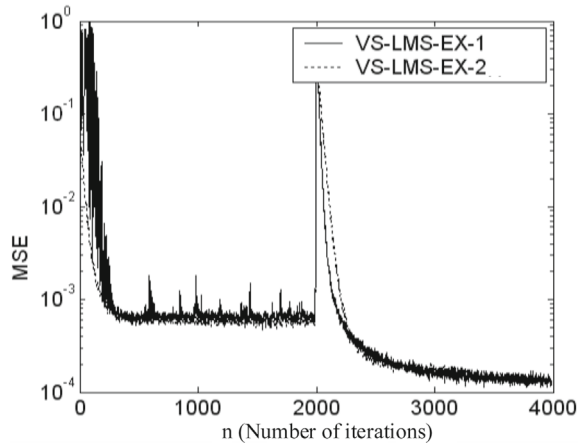


Fig. 8.37 Comparison of the two algorithms with optimal performance [29]

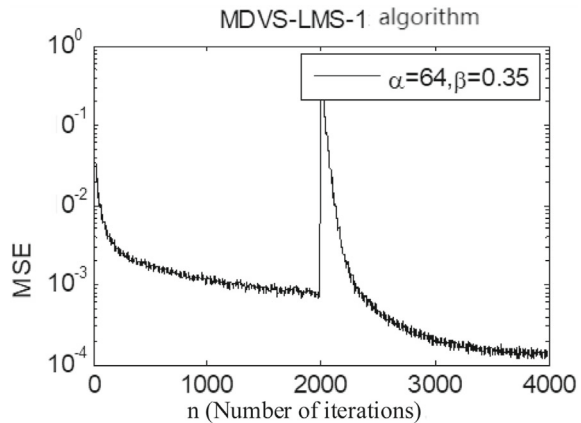


$$\mu(n+1) = \begin{cases} \mu_{\max} & \mu'(n+1) > \mu_{\max} \\ \mu_{\min} & \mu'(n+1) < \mu_{\min} \\ \mu'(n+1) & \text{else} \end{cases} \quad (8.116)$$

where $\mu_{\max} = 0.1$ and $\mu_{\min} = 0.005$.

The simulation is carried out according to the structure in Fig. 8.28. The channel characteristics change after 2000 points. Different experimental α and β affect the convergence speed, steady-state value of the mean squared error, and tracking speed. When $\alpha = 64$ and $\beta = 0.35$, the performance of the algorithm is the best. Five-hundred independent simulations of the MDVS-LMS-1 algorithm were conducted, and the set was averaged to obtain the mean squared error curve, as shown in Fig. 8.38.

Fig. 8.38 Mean squared error curve of the MDVS-LMS-1 algorithm [29]



iv. **MDVS-LMS-2 algorithm**

$$\mu(n) = \beta \left[1 - \frac{1}{\alpha \times |e(n) \times [e(n-1) + e(n)]| + 1} \right] \quad (8.117)$$

$$\mu(n+1) = \begin{cases} \mu_{\max} & \mu'(n+1) > \mu_{\max} \\ \mu_{\min} & \mu'(n+1) < \mu_{\min} \\ \mu'(n+1) & \text{else} \end{cases} \quad (8.118)$$

where $\mu_{\max} = 0.1$ and $\mu_{\min} = 0.005$.

The simulation is carried out according to the structure in Fig. 8.28. The channel characteristics change after 2000 points. Different experimental α, β affect the convergence speed, steady-state value of the mean squared error, and tracking speed. When $\alpha = 64$ and $\beta = 0.35$, the performance of the algorithm is the best. Five-hundred independent simulations of the MDVS-LMS-E-2 algorithm were conducted, and the set was averaged to obtain the mean squared error curve, as shown in Fig. 8.39

V. **VS-LMS-EE-1 algorithm**

$$p(n) = \beta p(n-1) + (1-\beta)e(n) \quad (8.119)$$

$$\mu(n+1) = \alpha \mu(n) + \gamma p(n) \quad (8.120)$$

$$\mu(n+1) = \begin{cases} \mu_{\max} & \mu(n+1) > \mu_{\max} \\ \mu_{\min} & \mu(n+1) < \mu_{\min} \\ \mu(n+1) & \text{else} \end{cases} \quad (8.121)$$

where $\mu_{\max} = 0.1$ and $\mu_{\min} = 0.005$.

Fig. 8.39 Comparison of the mean squared error curves of the MDVS-LMS-2 and DVS-LMS algorithms [29]

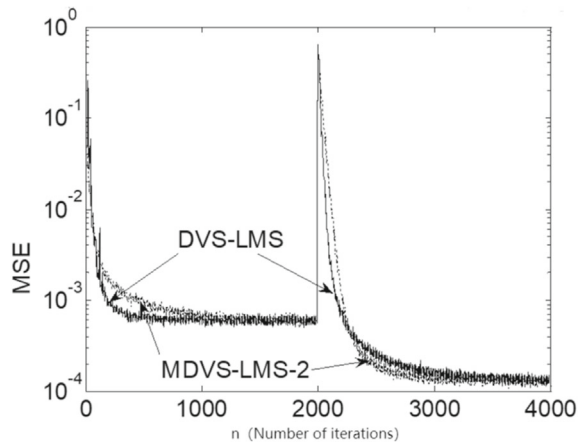
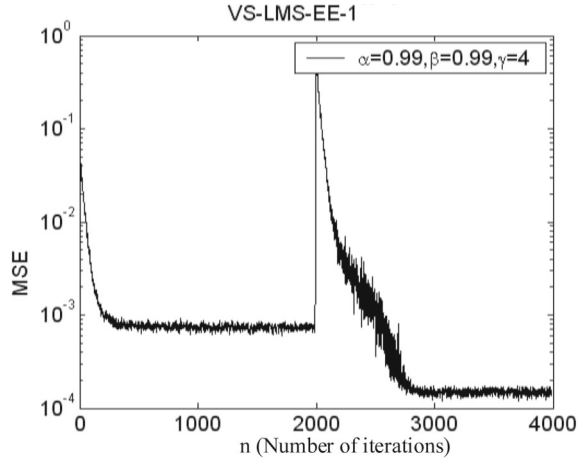


Fig. 8.40 Mean squared error curve of the VS-LMS-EE-1 algorithm [29]



The simulation is carried out according to the structure in Fig. 8.28. The channel characteristics change after 2000 points. Different experimental α , β affect the convergence speed, steady-state value of the mean squared error, and tracking speed. When $\alpha = 0.99$, $\beta = 0.99$, and $\gamma = 4$, the algorithm performance is the best. Five-hundred independent simulations of the VS-LMS-E-1 algorithm were conducted, and the set was averaged to obtain the mean squared error curve, as shown in Fig. 8.40.

vi. **VS-LMS-EE-2 algorithm**

$$\mu(n) = \beta(1 - \exp(-\alpha|e(n)e(n-1)|)) \quad (8.122)$$

The simulation is carried out according to the structure in Fig. 8.28. The channel characteristics change after 2000 points. Different experimental α and β affect the convergence speed, steady-state value of the mean squared error, and tracking speed. When $\alpha = 64$ and $\beta = 0.2$, the algorithm performance is the best. Five-hundred independent simulations of the VS-LMS-E-2 algorithm were conducted, and the set was averaged to obtain the mean squared error curve, as shown in Fig. 8.41.

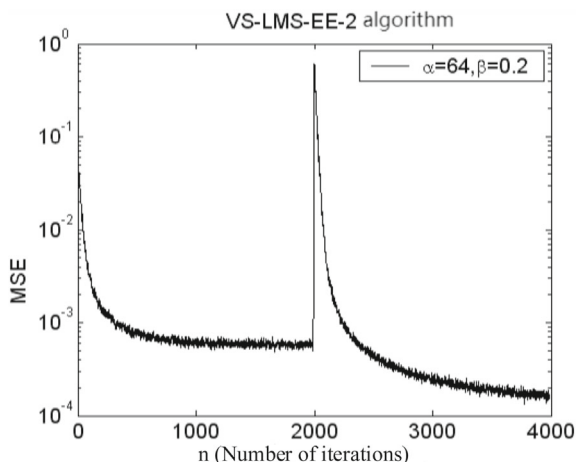
vii. **VS-LMS-EE-3 algorithm**

$$p(n) = \beta p(n-1) + (1 - \beta)e(n)e(n-1) \quad (8.123)$$

$$\mu(n+1) = \alpha\mu(n) + \gamma p^2(n) \quad (8.124)$$

The simulation is carried out according to the structure in Fig. 8.28. The channel characteristics change after 2000 points. Different experimental α and β affect the convergence speed, steady-state value of the mean squared error, and tracking speed.

Fig. 8.41 Mean squared error curve of the VS-LMS-EE-2 algorithm [29]



When $\alpha = 0.99$, $\beta = 0.1$, and $\gamma = 1$, the algorithm performance is the best. Five-hundred independent simulations of the VS-LMS-E-2 algorithm were conducted, and the set was averaged to obtain the mean squared error curve, as shown in Fig. 8.42.

Figure 8.43 shows a comparison of the above five algorithms with the best performance. As can be seen from the figure, they converge well, and the steady-state value of the mean squared error is small. Among them, the MDVS-LMS-1 and MDVS-LMS-2 algorithms proposed in Ref. [44] have the smallest steady-state value of the mean squared error; however, the latter has a faster tracking speed when the channel characteristics change and can converge to a smaller mean squared error.

Fig. 8.42 Mean squared error curve of the VS-LMS-EE-3 algorithm [29]

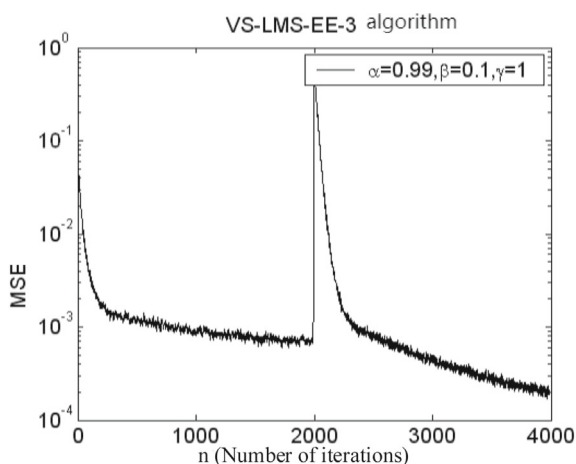
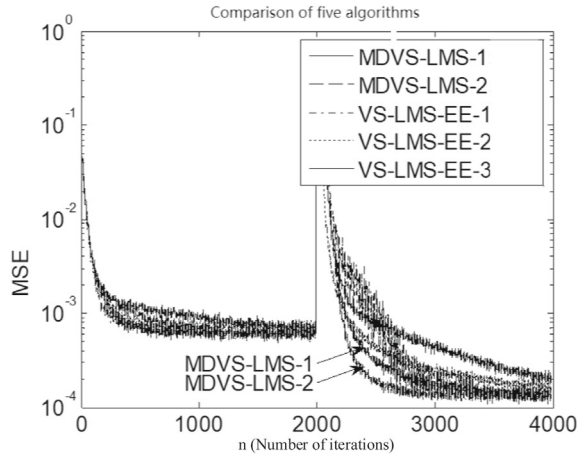


Fig. 8.43 Comparison of the five algorithms with the best performance [29]



In practical applications, it should be considered that the MDVS-LMS-1 and MDVS-LMS-2 algorithms proposed in Ref. [44] do not require exponential operations, and the number of operations is small (one iteration operation; the MDVS-LMS-1 algorithm needs $2N + 5$ multiplication and $2N + 3$ addition operations, and the MDVS-LMS-2 algorithm needs $2N + 5$ multiplication and $2N + 2$ addition operations), which allows them to easily realize real-time signal processing.

The channel changes after 2000 points. The number of bit errors in the MDVS-LMS-2 algorithm is observed. Decision threshold $gate = 0.5$ is selected, and the number of bit errors in this algorithm is shown in Fig. 8.44a and b.

As can be seen from Fig. 8.44a and b, before the channel characteristics change, the number of bit errors of the MDVS-LMS-2 algorithm is 0. After the channel characteristics change, the algorithm enters the tracking stage. Within 40 iterations, the filter coefficients converge again, and then there are no more bit errors; thus, demonstrating its good tracking performance.

(6) Simulation of a decision-guided equalizer based on a variable-step-size LMS algorithm [46]

The above simulation experiments of the variable-step-size LMS algorithm were to analyze and compare the performance of various algorithms. In the simulation process, the equalizer has been in the training stage. In practical applications, a decision-guided equalizer is used [47–49].

First, the equalizer enters the training stage to optimize the equalizer weight-vector coefficient as much as possible. Subsequently, equalizer $y(n)$ enters the working stage and uses the decision detector to directly generate the reference signal $d(n)$ from the equalizer output. At this time, the error signal $e(n)$ is the decision error. A block diagram of the adaptive-equalization experiment is shown in Fig. 8.45.

The information structure of the transmission sequence is shown in Fig. 8.46. In the experiment, the information length of the transmission sequence is 1024. The length of the training sequence is 64 and the length of the unknown user data is 960.

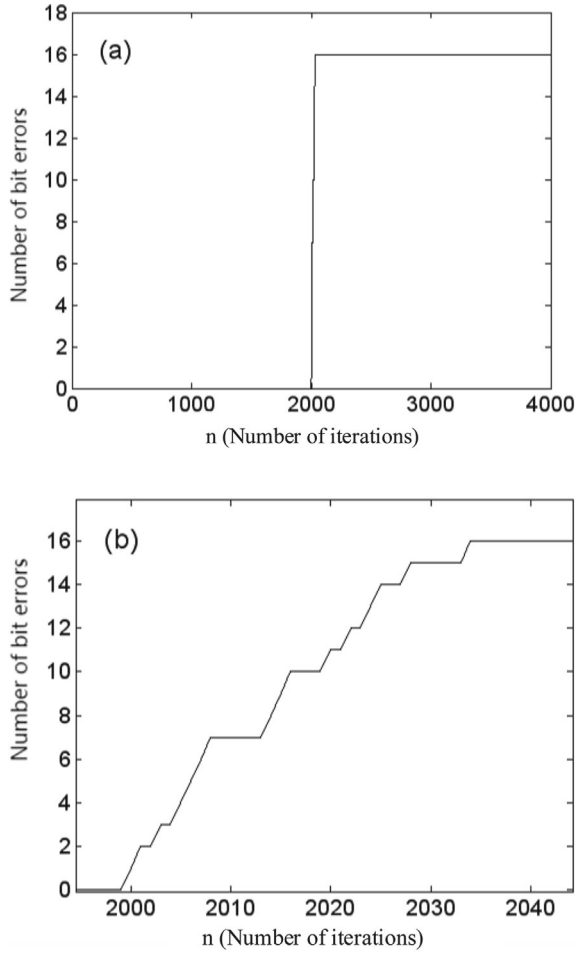


Fig. 8.44 Number of bit errors in the MDVS-LMS-2 algorithm **a** Number of algorithm bit errors [29] **b** Partial enlarged view [29]

The VS-LMS-E-3, VS-LMS-EX-2, and MDVS-LMS-2 algorithms proposed in this study have good equalization performance and a relatively small amount of computation. The simulations were carried out according to the block-diagram structure shown in Fig. 8.45. After 64 iterations, the decision-guided equalizer is adopted, and the channel characteristics change after 2000 points. The three algorithms were independently simulated 500 times, and the sets were averaged to obtain the mean squared error curves, as shown in Fig. 8.47.

As can be seen from Fig. 8.47, the convergence performance of the VS-LMS-E-3 and VS-LMS-EX-2 algorithms is the same, and the steady-state value of the mean squared error is little different, while the steady-state value of the mean squared error

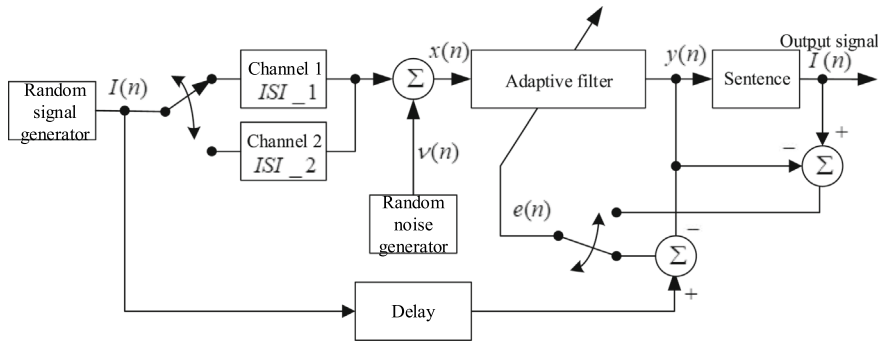
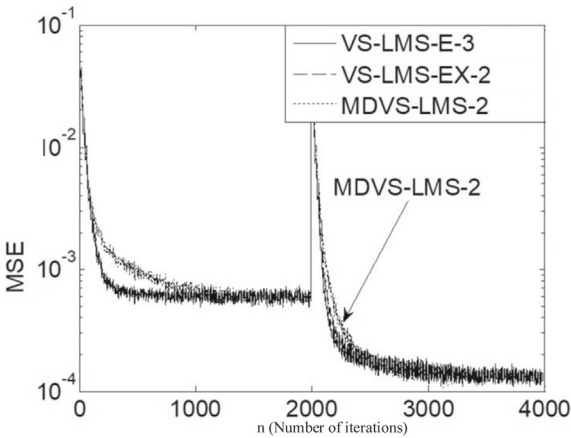


Fig. 8.45 Block diagram of adaptive-equalization experiment #3

Fig. 8.46 Information structure of the transmission sequence

Training sequence length 64	Unknown user-data length 960
-----------------------------	------------------------------

Fig. 8.47 Comparison of three algorithms when using a decision-guided equalizer [29]

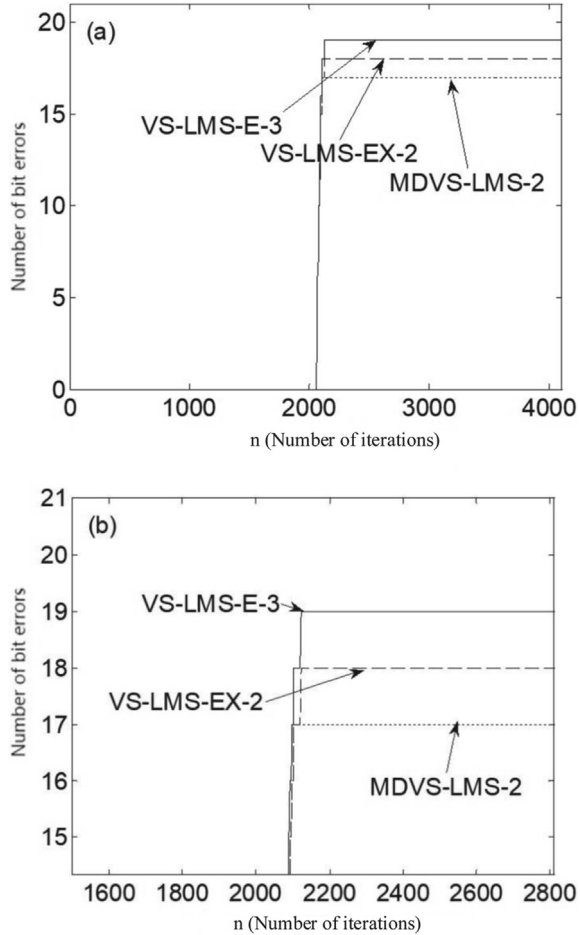


of the MDVS-LMS-2 algorithm is slightly larger. When the channel characteristics change, the tracking speed of the three algorithms is faster.

A comparison of the number of bit errors of the three algorithms applied in the decision-guidance equalizer is shown in Fig. 8.48a and b.

As can be seen from Fig. 8.48, when the decision-guided equalizer is adopted, the three algorithms do not produce bit errors before the channel characteristics change. After the channel characteristics change, the algorithms enter the tracking stage. After about 80 iterations, the algorithms converge again, and then there are no new bit errors, which shows good tracking performance. Among them, the MDVS-LMS-2 algorithm has the smallest number of errors and the VS-LMS-E-3 algorithm has the largest number of errors.

Fig. 8.48 Comparison of the bit error numbers of three algorithms **a** Comparison of the bit error numbers of three algorithms [29] **b** Partial enlarged view [29]



(7) Simulation of blind equalization

A simulated atmospheric laser channel is used for simulation. The transmission signal $I(n)$ is generated randomly. Then, it is modulated with attenuation coefficient σ , and Gaussian white noise v is added to obtain the filter input signal $x(n)$. The block diagram of the adaptive blind-equalization experiment is shown in Fig. 8.49. The order of the filter is 11 and the step factor is $\mu = 0.1$.

For the blind-equalization algorithm and blind-equalization algorithm + DD (decision guidance), environmental conditions in rainy days, moderately rainy days, light rainy days, foggy days, and cloudy days were simulated. Five-hundred independent simulations were conducted, respectively, and the sets were averaged to obtain the comparison of the mean squared error curves, as shown in Fig. 8.50a–e.

As can be seen from Fig. 8.50, under various weather conditions, the steady-state error of the blind-equalization algorithm is large, and the steady-state value of the

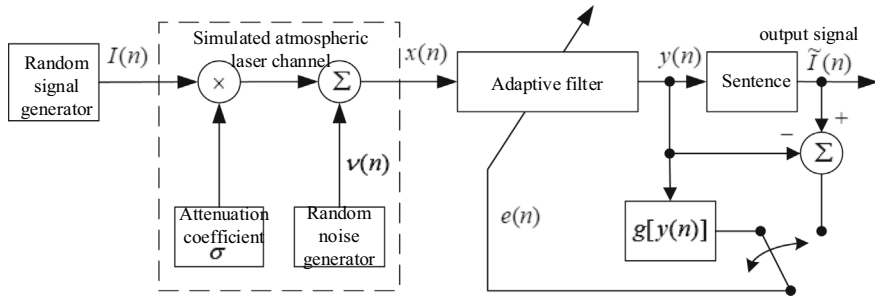


Fig. 8.49 Block diagram of the adaptive blind-equalization experiment [29]

large mean squared error is not conducive to signal judgment. The combination of blind equalization and decision guidance can greatly reduce the steady-state value of the mean squared error.

8.4.2 Analysis of Simulation Results

This chapter has studied the LMS algorithm, normalized LMS algorithm, simplified LMS algorithm, and variable-step-size LMS algorithm through simulation. Finally, the dual-mode blind-equalization algorithm was simulated.

It can be seen from these simulations of the LMS algorithm [29, 50], when the step size is large, the convergence speed is fast and the steady-state value of the mean squared error is also large. When the step size μ is small, the convergence speed slows down, but the steady-state value of the mean squared error also decreases. This reflects the contradiction between the convergence speed and the steady-state value of the mean squared error when the step size μ is fixed.

The convergence speed of the normalized LMS algorithm is faster than that of the LMS algorithm; however, the steady-state value of the mean squared error is relatively large. Among the three simplified LMS algorithms, simplified algorithm 2 converges quickly and the steady-state value of the mean squared error is small. The steady-state values of the mean squared error of simplified algorithms 1 and 3 are large.

Variable-step-size LMS algorithms can be roughly divided into three categories. The first type establishes the relationship between the step size and error signal. The four representative algorithms are the SVS-LMS, VS-LMS-E-1, DVS-LMS, and VS-LMS-E-3 algorithms. The four variable-step-size LMS algorithms converge well.

Among them, the steady-state value of the mean squared error of the SVS-LMS algorithm is the largest and its tracking speed is the slowest when the channel characteristics change. The steady-state value of the mean squared error of the other three

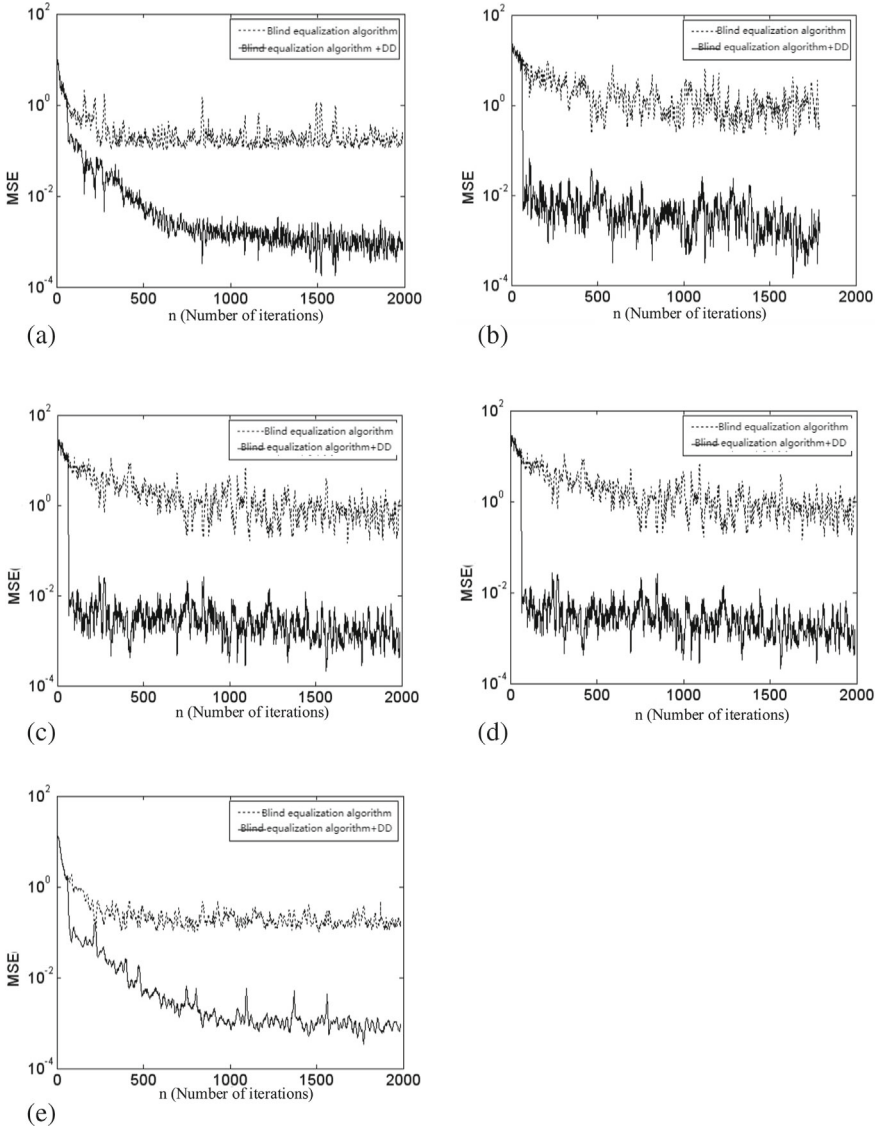


Fig. 8.50 Simulations of various weather conditions **a** Rainy days [29] **b** Moderately rainy days [29] **c** Light rainy days [29] **d** Foggy environment [29] **e** Cloudy environment [29]

algorithms is small and the tracking speed is fast, which reflects the performance improvement of these three algorithms over the SVS-LMS algorithm.

The SVS-LMS and VS-LMS-E-1 algorithms require exponential operations, and the number of operations is relatively large. The DVS-LMS algorithm is unstable in the initial stage. The VS-LMS-E-3 algorithm has a small steady-state value of the

mean squared error and a simple operation, which allows it to easily realize real-time signal processing.

The second type of variable-step-size algorithm establishes the relationship between the step size and the estimated value of the sum cross-correlation function. The two representative algorithms are the VS-LMS-EX-1 and VS-LMS-EX-2. Both algorithms converge well, and the steady-state values of the mean squared error are almost the same. However, in the initial stage, the mean squared error of VS-LMS-EX-1 is relatively unstable. When the channel characteristics change, the tracking speed of VS-LMS-EX-1 is faster. Among them, the computation of VS-LMS-EX-2 is relatively small.

The third algorithm type establishes the relationship between the step size $e(n)$ and the autocorrelation estimation value of the error signal $x(n)$. The five representative algorithms are MDVS-LMS-1, MDVS-LMS-2, VS-LMS-EE-1, VS-LMS-EE-2, and VS-LMS-EE-3, proposed in this paper. The five algorithms converge well, and their steady-state values of the mean squared error are small.

Among them, the MDVS-LMS-1 and MDVS-LMS-2 algorithms proposed in this paper have the smallest steady-state value of the mean squared error; however, the latter has a faster tracking speed when the channel characteristics change and converges to a smaller mean squared error. In practical applications, it should be considered that the MDVS-LMS-1 and MDVS-LMS-2 algorithms proposed in this paper do not require exponential operations and can easily realize real-time signal processing.

Among the three types of variable-step-size LMS algorithms, the algorithms with relatively good performance in each type include the VS-LMS-E-3, VS-LMS-EX-2, and MDVS-LMS-2 algorithms proposed in this paper. When the decision-guided equalizer is used, the equalization performance of the three algorithms is better, and the amount of computations is relatively small.

Through the simulation of the dual-mode blind-equalization algorithm, the steady-state value of the mean squared error of the blind-equalization algorithm is large, under various weather conditions. The combination of blind equalization and decision guidance can greatly reduce the steady-state value of the mean squared error and improve the algorithm performance.

References

1. Ke XZ, Xi XL (2004) Wireless laser communication. Beijing University of Posts and Telecommunications Press, Beijing
2. Zheng BY (2003) Principle of adaptive filter, 4th edn. Electronic Industry Press, Beijing
3. Dinis A (2004) Daptive filtering algorithm and implementation. Electronic Industry Press
4. Li Y (2003) Research on adaptive filtering and filtering algorithm. Northwest University of Technology
5. Deng JC, Shi BZ, Pei BN (2000) Exact solution of convergence step of LMS algorithm. Signal Process 3(1):50–54
6. Pei BN (1994) Convergence and step size selection of LMS algorithm. J Commun 15(4):106–111

7. Gong YH (2003) Adaptive filtering time domain adaptive filtering and smart antenna. Electronic Industry Press, Beijing
8. Wu GB, Zhu LY (1994) A variable step size LMS adaptive filtering algorithm. *Acta Electron Sin* 22(1):55–61
9. Qin JF, Wei G (1996) Variable step size LMS adaptive filtering algorithm based on S-type function. *Radio Engin* 4(5):44–47
10. Gao Y, Xie SL (2001) A variable step size LMS adaptive filtering algorithm and its analysis. *Acta Electron Sin* 8(9):1094–1097
11. Duan YB, Wang C (2004) A new variable step size least mean square algorithm. *J Daqing Petroleum Inst* 4(2):72–75
12. Chen K, Chen P (2004) A new variable step size LMS adaptive filtering algorithm. *Electron Technol* 23(6):56–58
13. Deng JB, Hou XG, Wu ZG (2004) Variable step size LMS adaptive algorithm based on skip tongue line. *Data Acquisit Process* 9(3):282–285
14. Kwong RH, Johnston EW (1992) A variable step size LMS algorithm. *IEEE Trans Signal Process* 40(7):76–85
15. Sheng SY, Wang JH (2002) A new variable step size LMS adaptive filtering algorithm. *J East China Inst Ship Build* 16(3):50–52
16. Yu Y, Y J S, Tian YF (2005) A new variable step size LMS algorithm and its simulation. *Gansu J Sci* 17(2):34–37
17. Li H, Li YB, Zou YP et al (2005) A new variable step size adaptive harmonic detection algorithm. *Power Syst Autom* 29(2):69–73
18. Xu K, Ji H, Le GX (2004) An improved variable step size adaptive filter LMS algorithm. *J Circuits Syst* 9(4):115–117
19. Tyseer A, Mayyas K (1997) A robust variable step-size LMS-type algorithm: analysis and simulations. *IEEE Trans Signal Process* 45(3):15–22
20. Jiang MF, Zheng XL, Peng CL (2001) A new variable step size LMS adaptive algorithm and its application in adaptive noise cancellation. *Signal Process* 17(3):282–286
21. Chen SZ, Li Y (2004) A new variable step size LMS adaptive algorithm and its application in adaptive cancellation. *J Jiangnan Univ* 3(6):583–586
22. Sristi P, Lu WS, Antoniou A (2001) A new variable-step-size LMS algorithm and its application in subband adaptive filtering for echo cancellation. *IEEE Int Symp* 2(36):721–724
23. Feng CQ, Zhang YS, Zhao ZB et al (2004) A new variable step size LMS adaptive algorithm and simulation. *Radio Engin* 34(4):31–33
24. Li YY, Wan JW, Zhou LZ (1999) An improved variable step size normalized LMS algorithm. *J Nat Univ Defense Technol* 21(1):94–96
25. Wang MQ, Hu GL, Zheng BY (2003) A new variable step size LMS adaptive filtering algorithm. *J Nanjing Univ Posts Telecommun* 23(4):12–16
26. Ma YH, Wang MQ (2004) A new variable step size LMS adaptive filtering algorithm and its algorithm analysis [J]. *Guangdong communication technology* 9(8):67–71
27. Wang MQ, Zheng BY (2004) A new variable step size LMS adaptive filtering algorithm. *Signal Process* 20(6):613–617
28. Li GM, Wang SP, Zeng ZH (2003) A new variable step size adaptive least mean square algorithm. *J Chang'an Univ* 23(4):88–90
29. Zhou XW (2006) Time domain equalization in atmospheric laser communication. Xi'an University of Technology, Xi'an
30. Zhou XW, Ke XZ (2006) A new variable step size LMS adaptive filtering algorithm. *J Xi'an Univ Technol* 45(3):290–293
31. Zhang XL, Ding JY, Zheng SX et al (2003) Research on variable order adaptive filter and its algorithm. *Modern Electron Technol* 36(16):39–42
32. Guo Y (2002) Constant modulus algorithm and its application in blind beamforming. Xi'an University of Electronic Science and technology, Xi'an
33. Feng J, Liao GS (2003) Constant modulus algorithm: progress and prospect. *Signal Process* 19(5):441–447

34. Zhu XG, Yang RZ, Zhu HW (2002) Comparative analysis and improvement of blind constant modulus equalization algorithms. *Commun Technol* 88(36):16–18
35. Zhao BF (2004) Research on variable step size blind equalization algorithm. Taiyuan University of Technology, Taiyuan
36. Pan LJ, Liu ZM (2004) Improved CMA + DD-LMS blind equalization algorithm. *Modern Wired Transmiss* 54(46):58–60
37. Dong X, Gong YH (2004) MIMO frequency selective channel adaptive equalization. *J Univ Electron Sci Technol China* 33(6):225–227
38. Pan LJ, Liu ZM (2005) A dual-mode blind equalization algorithm. *J Beijing Univ Posts Telecommun* 28(6):49–51
39. Wang F, Zhao JW, Li GJ (2002) Research on a blind equalization algorithm combining constant modulus and decision guidance. *J Commun* 23(6):105–109
40. Chen LX (2005) Experimental measurement of atmospheric laser communication system. Xi'an University of Technology, Xi'an
41. John GP (2005) Modern communication system, 2nd edn. Electronic Industry Press, Translated by Liu Shutang Beijing
42. Lei ZY, Hu GR, Sun LP (2003) Application of a new variable step size LMS algorithm in second-order adaptive notch filter. *J Shanghai Jiaotong Univ (Chin Ed)* 37(10):1564–1566
43. Liang H, Li ZS, Wang HG (2004) A variable step size adaptive lattice IIR Notch Filter and its application in reverberation cancellation. *Electroacoust Technol* 66(22):4–9
44. Zeng ZH, Wei L, Liu ST (1998) An improved transient step size LMS adaptive filtering algorithm. *J Electron Sci* 20(4):566–569
45. Lu ZY, Yang LW (1998) The first mock exam and performance analysis of a class of adaptive IIR filters. *J Appl Sci* 16(3):262–268
46. Zhang X (2003) Research on blind equalization algorithm based on Bussgang technology. Taiyuan University of Technology, Taiyuan
47. Yu FH, Petar M (2004) A blind particle filtering detector of signals transmitted over flat fading channels. *IEEE Trans Signal Process* 52(7):1891–1900
48. Oswaldo G, Silvestre R, Rafael P et al (2005) Error analysis of the simulated impulse response on indoor wireless optical channels using a monte Carlo-based ray-tracing algorithm. *IEEE Trans Commun* 53(1):124–130
49. Yujiro I, Ru WL (2002) A system-theoretic foundation for blind equalization of an FIR MIMO channel system. *IEEE Trans Circuits Syst* 49(4):425–436
50. Gu YT (2003) Research and application of convergence performance of LMS algorithm. Tsinghua University, Beijing

Chapter 9

Channel Blind Equalization Based on Higher-Order Statistics



Higher-order statistics, including higher-order moments, cumulants, moment spectra, and cumulant spectra, can be effectively used to analyze and process non-causal, non-minimum-phase, and non-Gaussian signals. In this chapter, the mathematical model of a Bussgang-like channel blind-equalization algorithm is discussed, according to the characteristics of optical wireless communications, with emphasis on the analysis of decision-directed methods combined with cumulant-based adaptive filtering. The algorithm is then simulated.

Several algorithms that directly use higher-order cumulants are investigated, including the odd-order (3, 2)-order normalization method, based on third-order cumulants, and the direct method, and the performance of these algorithms is analyzed. Considering the different signal attenuation of atmospheric laser-communication channels under different weather conditions, several typical algorithms are simulated using actual measurements of atmospheric laser-communication channels, and the performance of the algorithms is compared.

9.1 Blind-Equalization Algorithm

9.1.1 Adaptive and Blind Equalization

With the progress of digital-communication technology in terms of broadband, high speeds, and large capacity, adaptive-equalization technology has increasingly exposed its own shortcomings and defects, as follows.

Because the training sequence does not transmit useful information, it reduces the information-transmission rate in the communication system. For a fast time-varying channel, a training sequence must be periodically sent to update the channel-estimation parameters. Repeatedly sending the training sequence is costly, increases

the transmission overhead, and reduces the effective information-transmission rate and communication-system capacity.

In a communication system, strong interference in the channel or other factors may cause the equalizer to diverge at any time; then, the system must be retrained. Adaptive-equalization techniques are not self-recoverable and, to re-establish communication, it is necessary for the receiver to send a training sequence or wait until the next training sequence arrives to rebalance. Notifying the sender would require the system to use a feedback channel to provide timely feedback to determine whether retraining is required. However, this approach complicates the data-transmission system design and may be difficult to implement in practical applications.

In broadcast or point-to-multipoint communication networks, if the host is required to send another training sequence to rebalance a branch receiver, each time an unpredictable branch channel is added or a branch channel resumes work after a temporary failure, the additional traffic not only burdens the network but also interferes with the normal communication of other users sharing the channel resources. In addition, it may interrupt the host's communication with other branch channels.

In some applications, it is impossible to expect a training signal from the sender or to obtain a predetermined training sequence in all cases. For example, in information-interception and military-eavesdropping situations, it is clearly unrealistic to obtain an enemy-determined training signal. In signal-processing applications, such as seismic deconvolution and image reconstruction, where the sender is the natural world, it is also impossible to obtain an artificially set training signal. In wireless and mobile fading channels, the training sequence will be lost, owing to the effects of channel distortion.

For these reasons, it is necessary to investigate techniques that do not require the sender to send a known training sequence, but only to perform adaptive equalization based on the system's output observations. A technique based on adjusting the equalizer coefficients without using a training sequence is called blind equalization (BE).

The concept of "self-recovering equalization" (later called blind equalization) was first introduced in 1975 by Y. Sato, a Japanese scholar, as a simple improvement of the mean squared error function of traditional adaptive equalization [1]. Since then, scholars worldwide have devoted themselves to this research and have proposed a variety of blind-equalization algorithms using new mathematical theories and optimization methods, according to different application contexts.

Three main categories exist:

1. Bussgang algorithms, which are nonlinear memoryless transform functions at the output of an adaptive equalizer;
2. High-order or cyclic statistics algorithms that use high-order or cyclic statistics as mathematical tools;
3. Nonlinear equalizer algorithms, where the equalizer filter is nonlinear; i.e., nonlinear filters (e.g., Volterra filters) or neural networks, etc.

There are three classical forms of Bussgang algorithms: decision-directed, Sato, and Godard [2]. The decision-directed (DD) algorithm uses the judgment output as the desired signal to obtain a blind desired response. Sato's algorithm is the earliest Bussgang-type blind-equalization algorithm for privileged access-management (PAM) systems. This algorithm has good convergence capability for non-constant mode signals and the same convergence performance as the DD algorithm for constant-mode signals.

The beaconless geographic routing (BGR) algorithm was proposed by Benveniste in 1980 [3] and is an extension and application of the Sato algorithm for quadrature amplitude-modulation (QAM) systems. The Godard algorithm was proposed by Godard in 1980 and is based on adjusting the equalizer tap gain to minimize the cost function, which is constructed using the higher-order statistical characteristics of the transmitted signal [4].

Blind-equalization algorithms based on higher-order cumulative or cyclic statistics use higher-order or cyclic statistics as mathematical tools. The higher-order statistics (HOS) algorithm is a blind-equalization algorithm that uses the higher-order statistics of the channel-output signal. The algorithm can achieve global optimization, but it requires more sampling data and is computationally intensive.

Blind-equalization algorithms based on higher-order spectra (higher-order cumulants) were developed in the late 1980s [5]. The higher-order spectrum contains not only the amplitude but also the phase characteristics of a system or signal. Therefore, a relationship equation can be established between the higher-order accumulation of the signal and the channel parameters using only the received signal sequence, and the channel parameters can be obtained by solving the equation. In addition, because the higher-order accumulation of Gaussian noise is zero, higher-order accumulation and higher-order spectral estimation techniques can be used to extract non-Gaussian signals from Gaussian noise. Thus, higher-order accumulation can effectively detect and address nonlinear system problems.

Blind-equalization algorithms based on higher-order spectral theory are divided into two types: direct and indirect [6]. The direct method directly uses the higher-order accumulation of the received system sequence (i.e., the input sequence of the blind equalizer) to establish a relationship equation between the higher-order accumulation of the sequence and the channel parameters, which are then obtained by solving the equation. The indirect method first establishes a cost function containing a higher-order accumulation of the received sequence, then finds its extrema using an adaptive algorithm, and finally approximates the desired ideal system [7].

In 1980, Benveniste presented two important postulates about cumulants [3, 8]:

1. Second-order cumulants of a signal can only identify the amplitude-frequency characteristics of the system and cannot obtain phase information; therefore, higher-order cumulants must be utilized for non-minimum-phase systems.
2. For non-Gaussian signals, if the probability distribution of the system-output signal is the same as that of the input signal, the system is linear and distortion-free.

These two postulates constitute the theoretical basis for the study of blind equalization.

In 1991, Hatzinakos et al. first proposed a blind-equalization algorithm based on multiple spectra [9] (the tricepstrum equalization algorithm (TEA)), which could guarantee global convergence but was computationally intensive. In the same year, Porat et al. [10] proposed two blind-equalization algorithms for QAM systems, based on second- and fourth-order cumulants. In 1993, Zheng et al. proposed blind-equalization algorithms for PAM systems with second- and fourth-order cumulants. In 1996, Cadzow [11] first introduced the concept of normalized cumulants and proved that an equal magnitude of system input- and output-normalized cumulants is a sufficient and necessary condition for achieving blind equalization. Because the accumulation order is arbitrarily chosen, this sufficient condition is not one, but a cluster.

As higher-order statistical theory has developed in the directions of multidimensionality and spatialization, higher-order statistical methods in system identification and blind equalization are now developing in the direction of broader application areas and solving more complex problems. These are mainly manifested in the following aspects [12].

1. System identification in a non-Gaussian noise environment. Most present high-order statistic identification and equalization methods involve the study of non-Gaussian signals in Gaussian white-noise or colored-noise environments; however, there is less research on using higher-order statistics for system identification and equalization of non-Gaussian inputs in non-Gaussian additive colored-noise environments.
2. Using higher-order statistics to identify nonlinear models.
3. Development of cumulant multidimensionalization and pluralization to better solve the complex problems that exist in practice. Multidimensional signal accumulation provides more information than one-dimensional accumulation in model-sizing and parameter-estimation methods.

In addition, the complex form of non-Gaussian noise has various properties, such as the fourth-order cumulants being zero and satisfying the higher-order Yale–Walker equation. Therefore, the higher-order statistics of complex processes are also important for the study of asymmetric non-Gaussian signals of four orders or higher.

4. The combination of higher-order statistical methods and modern control theory has broad research and application prospects. However, there are still many limitations when using higher-order statistics methods for system identification. On the one hand, many current algorithms are theoretically derived and are difficult to apply in practice. On the other hand, the algorithms are relatively complex, and a large amount of calculations is the main reason for the current constraints of higher-order statistics methods, especially for solving real-time identification problems, which are difficult to achieve. In addition, the unique-identifiability

problem has not been solved well in some algorithms. To address these problems, the search for simple and applicable system-identification algorithms is currently a key theoretical and technical problem.

Blind-equalization techniques for higher-order statistics have been extensively developed, and the algorithms and theories have been extended and enriched from different perspectives and aspects. Wireless-communication technology has become a very active research hotspot in the communication field, where new techniques and methods are constantly emerging. Equalization is a key technology for achieving high speed and efficiency in wireless-communication systems. Therefore, it is necessary and urgent to study in depth the links between wireless-communication systems and their equalization techniques, and to continuously improve the performance of wireless-communication systems to meet the needs of various application areas.

Owing to the random nature of atmospheric channels in atmospheric laser-communication systems with adaptive-equalization techniques, a training sequence must be sent periodically to update the channel estimates. Owing to the strong interference and fading caused by an atmospheric channel as the weather changes, the equalizer may diverge at any time; thus, the system must be retrained.

Adaptive-equalization techniques are not self-recoverable; to re-establish communication, it is necessary for the receiver to notify the sender to send the training sequence immediately or to wait until the next training sequence arrives to re-equalize. The use of blind-equalization algorithms, which are self-recovering, allows the algorithm to reconverge. Therefore, it is important to study the application of blind-equalization algorithms in atmospheric laser communications [13].

9.1.2 Analysis of Blind-Equalization Algorithms for Higher-Order Statistics

Blind-equalization algorithms based on higher-order statistics can be classified into six methods: closed form, symmetric–antisymmetric transformation, direct, Shalvi and Weinstein (SW), normalization, and inverse-spectrum methods [14]. Some of the algorithms will be investigated in the following pages.

The transmit sequence $\{x(n)\}$ is assumed to be a non-Gaussian, independently and identically distributed (IID) smooth signal, and $\{y(n)\}$ is the channel output sequence [14]:

$$y(n) = x(n) * h(n). \quad (9.1)$$

In Eq. (9.1), $*$ is the convolution symbol, and $\{h(n)\}$ is the impulse function of the channel, which is also the input sequence of the equalizer. If channel $s(n)$ is the total system response, including the equalizer, the equalizer output sequence $\{\tilde{x}(n)\}$ is [14]

$$\tilde{x}(n) = x(n) * s(n), \quad (9.2)$$

where $*$ is the convolution symbol.

(1) Closed-form method

Giannakis first proposed the closed-form method in 1987 [15]. It illustrates that a system can be identified using the higher-order cumulants of the system output; this algorithm uses both autocorrelation and higher-order cumulants. Assuming that the system is a finite impulse response of order L , and setting $\gamma_{3x} = E\{x^3(k)\} \neq 0$, the relationship between the third-order cumulants and impulse response of the system is

$$c_{3y}(m, n) = \gamma_{3x} \sum_{i=0}^L h(i)h(i+m)h(i+n). \quad (9.3)$$

Assuming $h(0) = 1$, one can obtain

$$c_{3y}(L, k) = \gamma_{3x} h(L)h(k). \quad (9.4)$$

According to the symmetry formula,

$$c_{3y}(-L, -L, 0) = c_{3y}(L, 0) = \gamma_{3x} h(L). \quad (9.5)$$

Then,

$$h(k) = \frac{c_{3y}(L, k)}{c_{3y}(L, -L)} = \frac{c_{3y}(L, k)}{c_{3y}(L, 0)} \quad k = 0, 1, \dots, L, \quad (9.6)$$

where $c_{3y}(L, k)$ is a slice of third-order cumulant $c_{3y}(m, n)$, which is also commonly referred to as $c(L, k)$. Equations (9.5) and (9.6) yield an estimate of γ_{3x} based on the output observations:

$$\gamma_{3x} = \frac{c_{3y}(0, 0)}{\sum_{k=0}^L [c_{3y}(L, k)/c_{3y}(L, 0)]^3}. \quad (9.7)$$

Similarly, assuming a system of order Q for a fourth-order cumulant, the fourth-order cumulant can be expressed by the Barlett–Brillinger–Rosenblatt (BBR) formula as [14]

$$c_{4y}(l, m, n) = \gamma_{4x} \sum_{i=0}^L h(i)h(i+m)h(i+n). \quad (9.8)$$

Thus, we can obtain

$$h(k) = \frac{c_{4y}(L, L, k)}{c_{4y}(-L, -L, -L)} = \frac{c_{4y}(L, 0, k)}{c_{4y}(L, 0, 0)}, \quad (9.9)$$

where $c_{4y}(L, L, k)$ is a slice of $c_{4y}(l, m, n)$. Similarly, one can obtain

$$\gamma_{4x} = \frac{c_{4y}(0, 0, 0)}{\sum_{k=0}^Q [c_{4y}(L, 0, k)/c_{4y}(L, 0, 0)]^4}. \quad (9.10)$$

Equations (9.3)–(9.9) show that if the third- or fourth-order accumulation of the system output is estimated, the impulse response of the system can be obtained, and the system can be equalized. Moreover, the closed-form equation uses only one slice of the third- or fourth-order accumulation of the system output. Other slices can also be used because of their symmetry.

However, before using the closed-form method, it is necessary to assume that the order L or Q of the system is known or that the order can be determined from the cumulative output. In practice, it is not possible to obtain the exact higher-order accumulation but only to estimate it, based on a noisy observation. The closed-form method has no filtering to reduce the impact of cumulative-estimation errors; therefore, it is of academic interest only and lacks practical value.

In 1994, Zhang and Zhang [16] proposed a recursive-estimation method for closed formulas applicable to cumulants of order k that guarantees the uniqueness of the solution [17]. The algorithm mainly considers moving-average (MA) models based on the following form [14]:

$$h(n) = \sum_{i=0}^q b(i)\delta(n-i) = b(n). \quad (9.11)$$

Namely, $h(n) = b(n) = 0$ ($q < n < 0$).

Because $c_{ky}(\tau, 0, \dots, 0) = \gamma_{kx} \sum_{i=0}^q h^{k-1}(i)h(i+\tau)$, using the shorthand notation of [14]

$$c_{ky}(\tau, n) = c_{ky}(\tau, n, \dots, 0), \quad (9.12)$$

one can obtain

$$\gamma_{kx} \sum_{i=0}^q b(i)c_{ky}^{k-1}(q, i-\tau) = c_{ky}(\tau, 0)c_{ky}^{k-1}(q, 0). \quad (9.13)$$

Letting $\tau = -q$, $\tau = q$, and $\tau = q-1$ be substituted separately, the recursive formula for $b(i)$ is obtained by association. However, Mendel pointed out [57] that, although the present algorithm is theoretically feasible, computationally simple, and a unique solution exists, it does not have any smoothing or filtering effect on the

additive noise. In addition, owing to the nature of recursion, its closed-form solutions propagate errors. Consequently, they have not been widely used in practice [18].

(2) Direct method

The direct method obtains the channel parameters directly from estimating the higher-order accumulation of the received signal, establishing a relationship equation between the higher-order accumulation of the signal and the channel parameters, finding the channel parameters by solving the equation, and finally, establishing the equalizer transfer function. Figure 9.1 represents a principle block diagram of the algorithm for the direct method.

The direct method obtains the channel parameters by solving equations; thus, the global convergence of the algorithm is generally guaranteed, which is the greatest advantage of the direct method. In 1993, Zheng [19] found three linear systems of equations that illustrate the linear relationship between the second-, third-, and fourth-order accumulations of the signal and the equalizer coefficients, respectively. It was shown that when the channel is a linear time-invariant, causal, stable system with no zeros on the unit circle, the relationship between the equalizer tapping coefficient w_j and the channel impulse response h_i is

$$\frac{1}{\gamma_{2x}} \sum_{j=r_1}^{r_2} w_j c_{2y}(i+j) = h_i, \quad i \in [0, L] \quad (9.14)$$

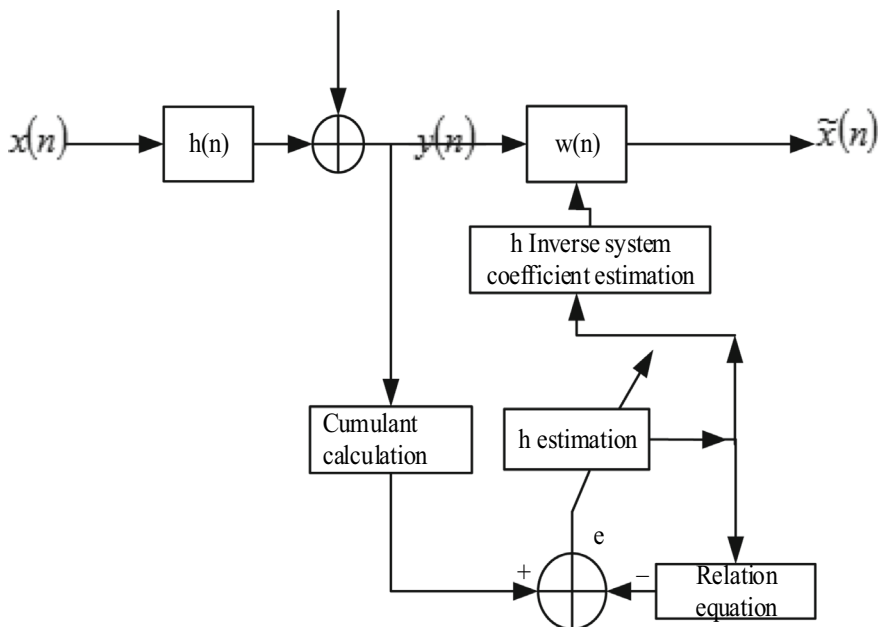


Fig. 9.1 Principle block diagram of the direct-method algorithm

$$\frac{1}{\gamma_{3x}} \sum_{j=r_1}^{r_2} w_j c_{3y}(i+j) = h_i^2, \quad i \in [0, L] \quad (9.15)$$

$$\frac{1}{\gamma_{4x}} \sum_{j=r_1}^{r_2} w_j c_{4y}(i+j) = h_i^3, \quad i \in [0, L], \quad (9.16)$$

where $\gamma_{4x} = \text{cum}(x^k(n))$ and $k = 2, 3, 4$; c_{2y} , c_{3y} , and c_{4y} are the second-, third-, and fourth-order accumulations of the equalizer input signal, respectively, L is the length of the transmission channel, and f is the order of the blind equalizer. Combining Eqs. (9.14) and (9.15) or Eqs. (9.14) and (9.16), h_i can be eliminated to obtain the tapping factor w_j of the equalizer, which determines its construction.

In 1993, Zheng [20] proposed an equalization algorithm using third-order accumulations. Assuming that $\hat{c}_{3y}(m, n)$ is the third-order accumulation of the equalizer input-sequence $y(n)$ estimate, the error function $\varepsilon = \hat{c}_{3y}(m, n) - c_{3y}(m, n)$ can then be constructed from Eq. (9.3). This error function is then used to construct the cost function of the algorithm, which can be obtained as follows:

$$\varepsilon^2 = \sum_{m=0}^L \sum_{n=0}^L [\hat{c}_{3y}(m, n) - c_{3y}(m, n)]^2. \quad (9.17)$$

To simplify the operation, which can be chosen, Eq. (9.17) can be simplified to

$$\varepsilon^2 = \sum_{m=0}^L \left[\hat{c}_{3y}(m, n) - \gamma_{3x} \sum_{i=0}^L h(i) h^2(i+m) \right]^2. \quad (9.18)$$

The cost function can be derived as follows:

$$\begin{aligned} \nabla(i) &= \frac{\partial \varepsilon^2}{\partial h(i)} \\ &= 2 \cdot \sum_{m=0}^L \left[\gamma_{3x} \cdot \sum_{l=0}^{L-m} h(l) \cdot h^2(l+m) - c_{3y}(m, m) \right] \\ &\quad \cdot (h(i+m) + 2 \cdot h(i) \cdot h(i-m)), \end{aligned} \quad (9.19)$$

where $i = 0, \dots, L$.

The steepest-descent method is then used to solve for the system parameters $h(i)$; that is, $h_{k+1}(i) = h_k(i) - \frac{1}{2} \mu(k) \cdot \nabla_i$, where $i = 0, 1, \dots, L$. The iteration step size is chosen to satisfy the algorithm-convergence conditions [20].

To expedite the algorithm convergence and ensure that a globally optimal solution is obtained, an approximate solution of $h(i)$ is found using a third-order cumulative closed-form solution, which is used as the initial value of the iterations [20]:

$$h(0) = \left[c_{3y}^2(L, 0) / c_{3y}^2(L, L) \right]^{1/3} \quad (9.20)$$

$$h(k) = h(0) \cdot \left[c_{3y}(L, k) / c_{3y}(L, 0) \right] \quad k = 1, 2, \dots, L. \quad (9.21)$$

In practical applications, the estimated value $\hat{c}_{3y}(m, n)$ is generally used to replace $c_{3y}(m, n)$; that is,

$$h(0) = \left[\hat{c}_{3y}^2(L, 0) / \hat{c}_{3y}^2(L, L) \right]^{1/3} \quad (9.22)$$

$$h(k) = h(0) \cdot \left[\hat{c}_{3y}(L, k) / \hat{c}_{3y}(L, 0) \right]. \quad (9.23)$$

In the above derivation, it is assumed that the system order is known; however, for an unknown system, the order must first be determined, using an autocorrelation function or a system-order estimation method with higher-order statistics, or by using an empirical judgment method [20, 21]. Zheng [20] proposed a method for estimating the system order using third-order cumulants.

First, assuming that the order of the MA model is J_m , the system order can be given by

$$S(J) = \sum_{i=0}^J c_{3y}^2(J, i), \quad J = 1, \dots, J_m, \quad (9.24)$$

where $J_m > L$. If there exists J_0 satisfying $J \geq J_0$ such that $S(J)$ reaches a minimum value $S(J_0)$, then the order L of the MA model is

$$L = J_0 - 1. \quad (9.25)$$

The inverse filter acts as an equalizer. After estimating the channel characteristics using the above algorithm with third-order cumulants, the coefficients of the inverse filter [20], that is, the coefficients of the equalizer, can be found. The coefficients of the inverse filter are determined as follows [20], where $H(z)$ is the estimated z -domain form of the channel characteristics and $H^{-1}(z)$ is the z -domain form of the inverse-filter system function:

$$H^{-1}(z) = \frac{1}{H(z)} = \frac{1}{\sum_{i=0}^q b_i z^{-i}} = \sum_{i=r_1}^{r_2} \theta_i z^{-i}, \quad (9.26)$$

where r_1 is the order of the non-causal part of the inverse-system function, and r_2 is the order of the causal part of the inverse-system function.

Let $H^{-1}(z) = \sum_{i=1_1}^q \frac{A_i}{1 - z_i z^{-1}}$ be such that

$$|z_{m1}|, |z_{m2}|, \dots, |z_{ms}| < 1, \quad |z_{n1}|, |z_{n2}|, \dots, |z_{nt}| > 1, \quad (9.27)$$

where $s + t = q$. Assume that $|z_{mM}| = \max_{i=1, \dots, s} \{|z_{mi}|\}$. Then, it follows that

$$r_2 = [\ln \zeta / \ln |z_{mM}|] + 1, \quad (9.28)$$

where $\zeta < 1$. If no z_{mi} satisfies $|z_{mi}| < 1$, then $r_2 = -1$. Similarly, it follows that

$$r_1 = [\ln \zeta / \ln |z_{nN}|] - 1 \quad (9.29)$$

$$|z_{mM}| = \max_{i=1, \dots, s} \{|z_{ni}|\}, \quad |z_{ni}|, i = 1, \dots, t. \quad (9.30)$$

If no z_{ni} satisfies $|z_{ni}| < 1$, then $r_1 = 0$. Using the steepest-descent method, θ_i can be expressed as follows:

$$\varepsilon^2 = \sum_{l=0}^L \left[\sum_{i=r_1}^{r_2} \theta_i g_l^{-i} - H_R^{-1}(\alpha_l) \right]^2, \quad (9.31)$$

where $g_l = e^{jal}$, $l = 0, \dots, L$, $L \geq (r_2 - r_1 + 1)$.

(3) Normalization algorithm

A normalized cumulant is the ratio of two higher-order cumulants of different orders (one of which is normalized by the other). The basic-principle block diagram is shown in Fig. 9.2.

In Fig. 9.2, $\{x(n)\}$ is the transmit sequence, $\{\tilde{x}(n)\}$ is the output sequence of the blind equalizer, and the cumulants of order N are $c_{N\tilde{x}}$ and $c_{N\tilde{x}}$. They can be obtained from the BBR formula for higher-order cumulants [14]:

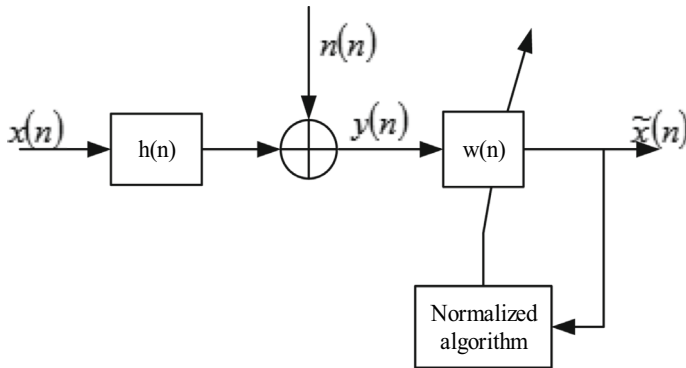


Fig. 9.2 Principle block diagram of the normalized-equalization algorithm

$$c_{N\bar{x}} = c_{Nx} \sum_L \{s_L^N(n)\}, \quad (9.32)$$

where $\{s_L(n)\}$ denotes the total response of the channel. We define the (M, N) -order normalized cumulant of $\bar{x}(n)$ as

$$k_{\bar{x}}(M, N) = \frac{c_{M\bar{x}}}{|c_{N\bar{x}}|^{M/N}}. \quad (9.33)$$

Assuming $c_{N\bar{x}} \neq 0$, one can similarly obtain $k_x(M, N)$. Using Equation $c_{N\bar{x}} = c_{Nx} \sum_L \{s_L^N(n)\}$, one obtains [14]

$$k_{\bar{x}}(M, N) = \frac{\sum_L s_L^M(n)}{\left| \sum_L s_L^N(n) \right|^{M/N}} k_x(M, N). \quad (9.34)$$

It can be proved that when $M > N$, there must be $\left| \sum_n s_L^N(n) \right|^{M/N} \geq \sum_n s_L^M(n)$. A necessary and sufficient condition for the equal sign of the inequality to be true is that there is only one nonzero element in $s_L(n)$. This indicates that a necessary and sufficient condition for the equal sign of $|k_{\bar{x}}(M, N)| \leq |k_x(M, N)|$ to be true is that $s_L(n)$ is set to zero.

Similarly, when $M < N$ and M is even, a necessary and sufficient condition for the equal sign of $\left| \sum_n s_L^N(n) \right|^{M/N} \leq \sum_n s_L^M(n)$ to be true is that there is only one nonzero element in $\{s_L(n)\}$. This leads to the conclusion that the necessary and sufficient condition for the equal sign to be true for $|k_{\bar{x}}(M, N)| \geq |k_x(M, N)|$ also satisfies the zero-setting condition for $\{s_L(n)\}$. Thus, the equalization problem can be translated into a channel-maximization method.

If we set $M = 3$ and $N = 2$, it follows from the theorem that the normalized cumulant of the channel output is always less than or equal to the normalized cumulant of the channel input; that is, $|k_{\bar{x}}(3, 2)|$ has an upper bound. In this manner, the blind-equalization problem is transformed into an unconstrained maximization problem of the normalized cumulant of the channel output. This maximization problem can be found using the stochastic gradient-descent algorithm [14].

Cadzow's theory implies that the blind-equalization problem can be solved using both higher-order cumulants simultaneously. However, Cadzow's polarization method is based on real channels and real signals, and it is possible to generalize this approach to complex channel and complex signal cases.

9.1.3 Normalized Performance Simulation

For atmospheric laser-communication channel characteristics, the noise affecting the signal transmission is mainly divided into two types: multiplicative and additive. Therefore, we divide the simulation environment into three cases to examine the performance of the (3, 2)-order normalized cumulative-equalization algorithm for the following three channel models: additive noise, multiplicative noise, and both additive and multiplicative noise.

In the simulation experiments, we used a randomly generated binary sequence as the transmit signal. The transmit signal $\{x(n)\}$ is passed through the simulated channel to obtain the equalizer in/out signal $\{y(n)\}$ and the equalizer initial tap-power vector $w(0) = (0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0)$.

Because multiplicative channel noise can cause intercode interference in signal transmissions, without loss of generality, the following constant reference channel ISI characteristics are used to reflect this problem in this study [22]:

$$\text{ISI} = (0.05, -0.063, 0.088, -0.126, -0.25, 0.9047, 0.25, 0, 0.126, 0.038). \quad (9.35)$$

For additive noise, we use additive Gaussian white noise (AGWN) with a mean of 0 and variance 0.01 ν . According to Eq. (9.33), the (3, 2)-order normalized cumulants of the transmit sequence $\{x(n)\}$ and the blind-equalizer output sequence $\{\tilde{x}(n)\}$ can be obtained as [23]

$$k_x(3, 2) = \frac{c_{3\tilde{x}}}{|c_{2x}|^{3/2}} \quad \text{and} \quad k_{\tilde{x}}(3, 2) = \frac{c_{3\tilde{x}}}{|c_{2\tilde{x}}|^{3/2}}. \quad (9.36)$$

The BBR formula for higher-order cumulants yields [23]

$$c_{2\tilde{x}} = c_{2x} \sum_L \{s_L^2(n)\} \quad (9.37)$$

$$c_{3\tilde{x}} = c_{3x} \sum_L \{s_L^3(n)\}, \quad (9.38)$$

where $\{s_L(n)\}$ denotes the total response of the channel. We define the (3, 2)-order normalized cumulant of $\tilde{x}(n)$ as [23]

$$k_{\tilde{x}}(3, 2) = \frac{c_{3\tilde{x}}}{|c_{2\tilde{x}}|^{3/2}}. \quad (9.39)$$

Assuming that $c_{2\tilde{x}} \neq 0$, the same can be obtained for $k_x(3, 2)$. Using $c_{N\tilde{x}} = c_{N\tilde{x}} \sum_L \{s_L^N(n)\}$, the following can be obtained from Eq. 9.34 [23]:

$$k_{\tilde{x}}(3, 2) = \frac{\sum_L s_L^3(n)}{\left| \sum_L s_L^2(n) \right|^{3/2}} k_x(3, 2). \quad (9.40)$$

Defining the cost function as $J(n) = k_{\tilde{x}}(3, 2)$, the following algorithm was obtained using the steepest-descent method [23]:

$$W(n+1) = W(n) - \mu \frac{\partial J(n)}{\partial W(n)}. \quad (9.41)$$

Because

$$k_{\tilde{x}}(3, 2) = \frac{c_{3\tilde{x}}}{(c_{2\tilde{x}})^{3/2}} = \frac{E[\tilde{x}^3] - 3E[\tilde{x}] \cdot E[\tilde{x}^2] + 2(E[\tilde{x}^2])^3}{[E[\tilde{x}^2] - (E[\tilde{x}])^2]^{3/2}}, \quad (9.42)$$

the partial derivative is obtained:

$$\begin{aligned} \frac{\partial J(n)}{\partial W(n)} &= \frac{\partial k_{\tilde{x}}(3, 2)}{\partial W(n)} \\ &= 3 \left(\frac{E[\tilde{x}^2(n)E[\tilde{x}^2(n)]Y(n) - \tilde{x}^2(n)\{E[\tilde{x}^2(n)]\}^3Y(n)}{[E[\tilde{x}^2] - (E[\tilde{x}])^2]^{5/2}} \right. \\ &\quad - \frac{4\tilde{x}(n)\{E[\tilde{x}^2(n)]\}^2\{E[\tilde{x}^2(n)]\}^2Y(n)}{[E[\tilde{x}^2] - (E[\tilde{x}])^2]^{5/2}} \\ &\quad \left. + \frac{2\tilde{x}(n)\{E[\tilde{x}^2(n)]\}^3Y(n) + 2\tilde{x}(n)\{E[\tilde{x}(n)]\}^3Y(n)}{[E[\tilde{x}^2] - (E[\tilde{x}])^2]^{5/2}} \right) \\ &\quad + 3 \left(\frac{E[\tilde{x}(n)E[\tilde{x}(n)]E[\tilde{x}^2(n)]Y(n) - \tilde{x}(n)E[\tilde{x}^3(n)]Y(n)]}{[E[\tilde{x}^2] - (E[\tilde{x}])^2]^{5/2}} \right. \\ &\quad + \frac{2\tilde{x}(n)\{E[\tilde{x}^2(n)]\}^3Y(n) - 2E[\tilde{x}^2(n)]\{E[\tilde{x}(n)]\}^2Y(n)}{[E[\tilde{x}^2] - (E[\tilde{x}])^2]^{5/2}} \\ &\quad \left. + \frac{E[\tilde{x}(n)]E[\tilde{x}^3(n)]Y(n) - \{E[\tilde{x}^2(n)]\}^3Y(n)}{[E[\tilde{x}^2] - (E[\tilde{x}])^2]^{5/2}} \right), \quad (9.43) \\ \frac{\partial J(n)}{\partial W(n)} &= \frac{\partial k_{\tilde{x}}(3, 2)}{\partial W(n)} = 3 \frac{E[\tilde{x}^2(n)[E[\tilde{x}^2(n)] - \{E[\tilde{x}^2(n)]\}^3]Y(n)}{[E[\tilde{x}^2] - (E[\tilde{x}])^2]^{5/2}} \\ &\quad + 3 \frac{E[2E[\tilde{x}(n)]\{E[\tilde{x}^2(n)]\}^3 - 2E[\tilde{x}^2(n)]\{E[\tilde{x}(n)]\}^2 + E[\tilde{x}(n)]E[\tilde{x}^3(n)] - \{E[\tilde{x}^2(n)]\}^3]Y(n)}{[E[\tilde{x}^2] - (E[\tilde{x}])^2]^{5/2}} \end{aligned}$$

$$-3 \frac{E[\tilde{x}(n) \left[4\{E[\tilde{x}^2(n)]\}^2 \{E[\tilde{x}^2(n)]\}^2 - 2\{E[\tilde{x}^2(n)]\}^3 - 2\{E[\tilde{x}(n)]\}^3 - E[\tilde{x}(n)]E[\tilde{x}^2(n)] + E[\tilde{x}^3(n)] \right] Y(n)]}{[E[\tilde{x}^2] - (E[\tilde{x}])^2]^{5/2}} \quad (9.44)$$

Applying the stochastic gradient estimate to $W(n+1) = W(n) - \mu \frac{\partial J(n)}{\partial W(n)}$ gives the following iterative formula:

$$\begin{aligned} W(n+1) = W(n) - 3\mu & \left(\frac{\tilde{x}^2(n)\langle\tilde{x}^2(n)\rangle - \tilde{x}^2(n)\langle\tilde{x}^2(n)\rangle^3 - 4\tilde{x}(n)\langle\tilde{x}^2(n)\rangle^2\langle\tilde{x}^2(n)\rangle^2}{[\langle\tilde{x}^2(n)\rangle - \langle\tilde{x}(n)\rangle^2]^{5/2}} \right. \\ & + \frac{2\tilde{x}(n)\langle\tilde{x}^2(n)\rangle^3 + 2\tilde{x}(n)\langle\tilde{x}(n)\rangle^3 + \tilde{x}(n)\langle\tilde{x}(n)\rangle\langle\tilde{x}^2(n)\rangle - \tilde{x}(n)\langle\tilde{x}^3(n)\rangle}{[\langle\tilde{x}^2(n)\rangle - \langle\tilde{x}(n)\rangle^2]^{5/2}} \\ & \left. + \frac{2\langle\tilde{x}(n)\rangle\langle\tilde{x}^2(n)\rangle^3 - 2\langle\tilde{x}^2(n)\rangle\langle\tilde{x}(n)\rangle^2 + \langle\tilde{x}(n)\rangle\langle\tilde{x}^3(n)\rangle - \langle\tilde{x}^2(n)\rangle^3}{[\langle\tilde{x}^2(n)\rangle - \langle\tilde{x}(n)\rangle^2]^{5/2}} \right) Y(n), \end{aligned} \quad (9.45)$$

where μ is the step factor, and $\langle\tilde{x}^2(n)\rangle$ and $\langle\tilde{x}^3(n)\rangle$ are the estimates of $E[\tilde{x}^2(n)]$ and $E[\tilde{x}^3(n)]$, respectively; that is,

$$\langle\tilde{x}^3(n+1)\rangle = \rho\langle\tilde{x}^3(n)\rangle + (1-\rho)\tilde{x}^3(n) \quad (9.46)$$

$$\langle\tilde{x}^2(n+1)\rangle = \delta\langle\tilde{x}^2(n)\rangle + (1-\delta)\tilde{x}^2(n) \quad (9.47)$$

$$\langle\tilde{x}^2(n+1)\rangle - \langle\tilde{x}(n+1)\rangle = \alpha(\langle\tilde{x}^2(n)\rangle - \langle\tilde{x}(n)\rangle) + (1-\alpha)(\langle\tilde{x}^2(n)\rangle - \langle\tilde{x}(n)\rangle) \quad (9.48)$$

$$\langle\tilde{x}(n+1)\rangle = \beta\langle\tilde{x}(n)\rangle + (1-\beta)\tilde{x}(n), \quad (9.49)$$

where $0 \leq \rho \leq 1$, $0 \leq \delta \leq 1$, $0 \leq \alpha \leq 1$, and $0 \leq \beta \leq 1$ are their respective forgetting factors.

(1) Additive noise

A schematic diagram of the channel model with only additive noise interference is shown in Fig. 9.3.

In Fig. 9.3, the transmit signal $\{x(n)\}$ is passed through the additive noise v to obtain the normalized blind-equalizer input signal $\{y(n)\}$. The block diagram of the normalized blind-equalization algorithm experiment is shown in Fig. 9.4. The simulations were conducted according to this structure. In this study, 500 independent simulations were conducted for different step sizes by applying the normalization algorithm and using the error-squared magnitude estimate between the transmit sequence and the equalized output with the number of iterations n to obtain a statistical mean squared error (MSE) curve.

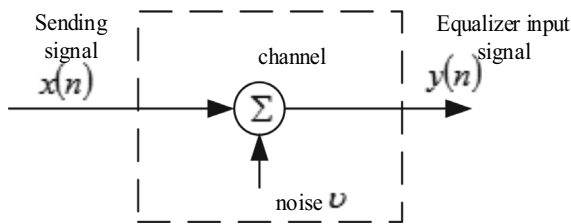


Fig. 9.3 Channel model 1

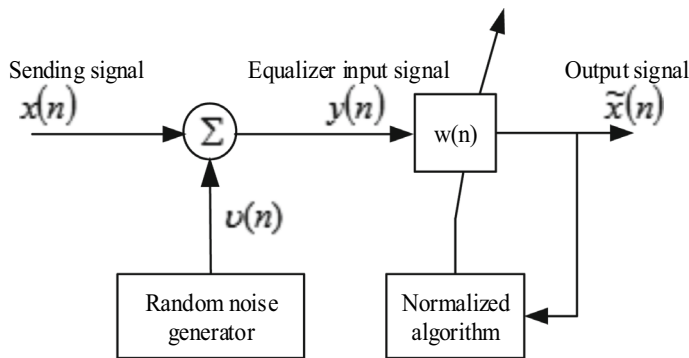


Fig. 9.4 Experimental block diagram 3 of the normalized blind-equalization algorithm

The step size is set to $stepsize = 0.0001$, $stepsize = 0.1 \times 10^{-5}$, and $stepsize = 0.1 \times 10^{-7}$, and the parameters are set as shown in Table 9.1. The MSE curve of the equalizer can be plotted according to the simulation of this algorithm, as shown in Fig. 9.5. In Fig. 9.5, when $stepsize$ is larger than 0.0001, the normalized-equalization algorithm converges faster; however, the steady-state error is larger. When $stepsize$ is smaller than 0.0001, the normalized-equalization algorithm converges more slowly; however, the steady-state error decreases.

(2) Multiplicative noise

The channel model for the case with only multiplicative noise is shown schematically in Fig. 9.6.

In Fig. 9.6, the transmit signal $\{x(n)\}$ is passed through the constant reference channel ISI to obtain the normalized blind-equalizer in/out signal $\{y(n)\}$. The block diagram of the normalized blind-equalization algorithm experiments is shown in Fig. 9.7. The simulations were carried out according to this structure. Five hundred independent simulations were applied to the normalization algorithm to obtain a

Table 9.1 Algorithm parameter selection for only additive noise

ρ	δ	α	β
0.996	0.9971	0.9985	0.997

Fig. 9.5 MSE curve of the normalized method with different step sizes

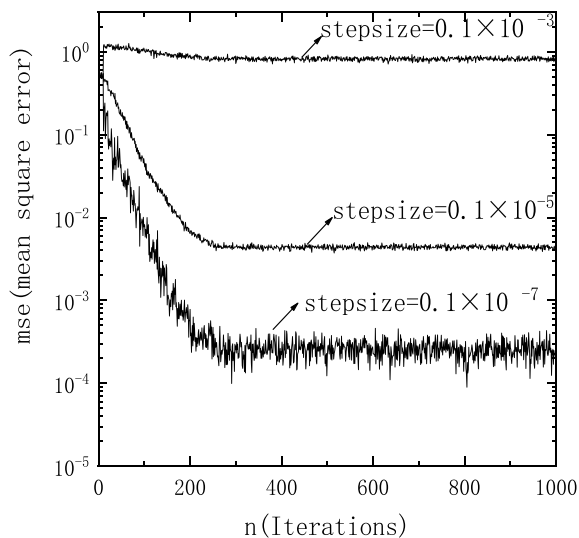
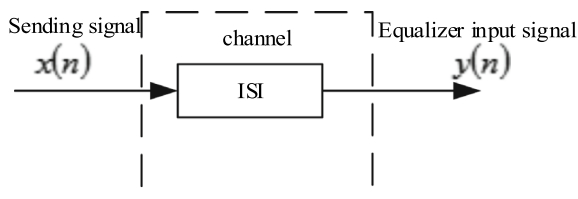


Fig. 9.6 Channel model with only multiplicative noise



statistically averaged MSE curve. Figure 9.7 indicates that the step size was set as $stepsize = 0.1 \times 10^{-7}$ and the parameters were as shown in Table 9.2.

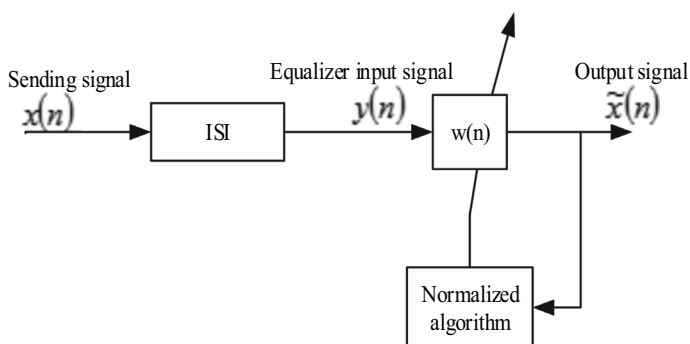
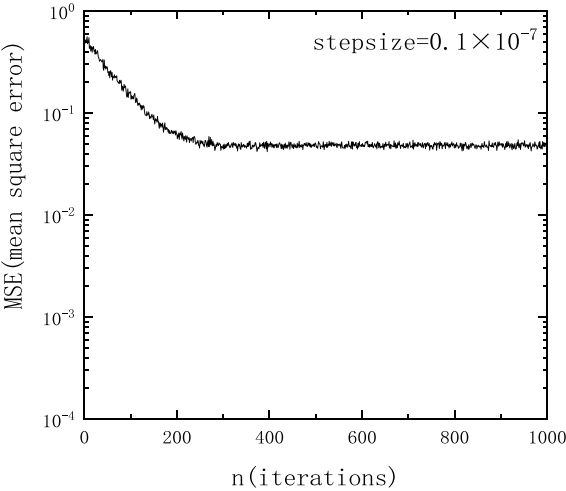


Fig. 9.7 Experimental block diagram of normalized blind equalization algorithm with multiplicative noise

Table 9.2 Algorithm parameter selection in the case with only multiplicative noise

ρ	δ	α	β
0.997	0.9978	0.9989	0.9977

Fig. 9.8 Algorithm simulation of mean square error (MSE) curve under multiplicative noise



The MSE curves for the algorithm simulation are shown in Fig. 9.8. It shows that the steady-state error of the normalized-equalization algorithm is larger when the channel model has only multiplicative noise.

Comparing the performance of the normalized blind-equalization algorithm when applied to the two channel models, as shown in Fig. 9.9, the convergence rate of the normalized method is lower than that when applied to the channel model with only additive noise, when the step size is fixed at $stepsize = 0.1 \times 10^{-7}$; however, the steady-state error is significantly higher.

(3) Multiplicative and additive noise

The channel model with multiplicative and additive noise is illustrated in Fig. 9.10.

In Fig. 9.10, the transmit signal $\{x(n)\}$ is attenuated by the constant reference channel ISI , and then background-noise interference is added to obtain the normalized blind-equalizer in/out signal $\{y(n)\}$. The block diagram of the normalized blind-equalization algorithm experiment is shown in Fig. 9.11.

The simulations was carried out according to this structure, and the statistical average MSE curve was obtained by applying the normalization algorithm for 500 independent simulations. Figure 9.12 represents the MSE curve when the step size is set as $stepsize = 0.1 \times 10^{-7}$. The parameters were set as listed in Table 9.3.

As can be seen in Fig. 9.12, the steady-state error of the normalized-equalization algorithm is also relatively large when the channel model has additive and multiplicative noise.

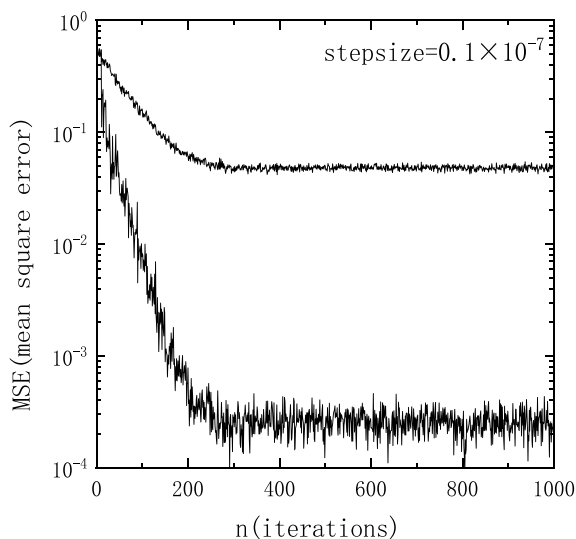


Fig. 9.9 Performance comparison of two normalized blind equalization algorithm channel models

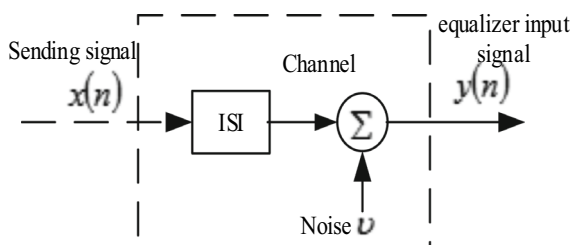


Fig. 9.10 Channel model with multiplicative and additive noise

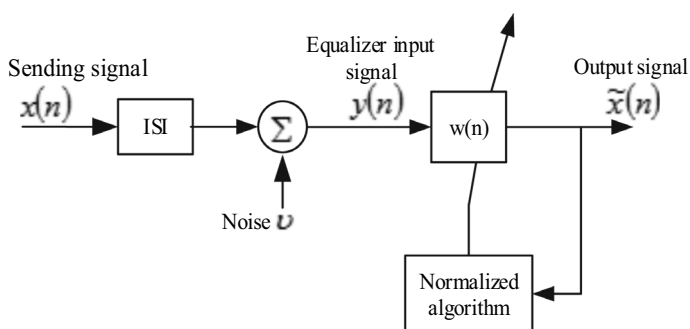


Fig. 9.11 Experimental block diagram of normalized blind equalization algorithm with multiplicative noise and additive noise

Fig. 9.12 Algorithm simulation of mean square error (MSE) curve under multiplicative noise and additive noise

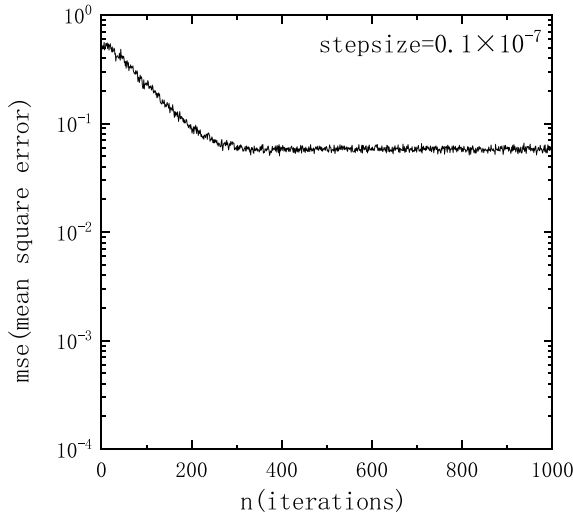


Table 9.3 Algorithm parameter selection in the case of both multiplicative and additive noise

ρ	δ	α	β
0.9972	0.998	0.9987	0.9979

The mean squared error curves comparing the normalized blind-equalization algorithm applied to the above three channel models are shown in Fig. 9.13. When the step size is fixed at $stepsize = 0.1 \times 10^{-7}$, the convergence rate of the normalized algorithm applied to the channel models in Figs. 9.7 and 9.11 is lower than that applied to the channel model in Fig. 9.4, and the steady-state error is significantly higher. Furthermore, the difference in the steady-state error when applied to the channel models of Figs. 9.7 and 9.11 is not significant.

9.1.4 Direct-Method Simulation Results

In the simulation experiments, to set the initial tapped-power vector of the equalizer, the direct blind-equalization algorithm first finds the initial iterative value of the channel characteristics using a closed-form solution. It then finds the inverse system function, which is the initial power vector of the equalizer.

(1) Additive noise

The direct blind-equalization algorithm was applied in a simulation environment with only additive noise, and the block diagram of the simulation experiment is shown in Fig. 9.14.

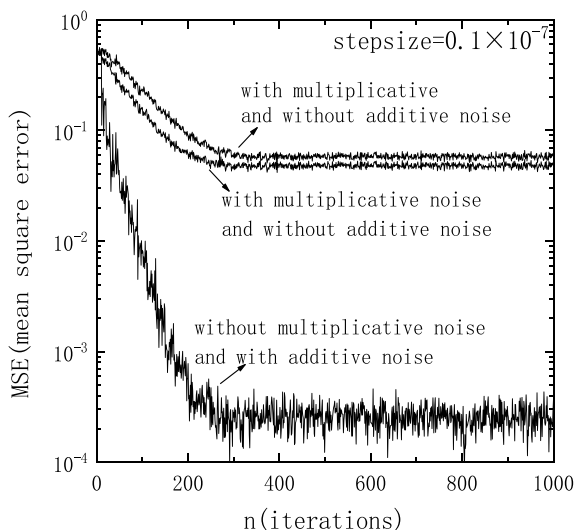


Fig. 9.13 Performance comparison of three normalized blind equalization algorithm channel models

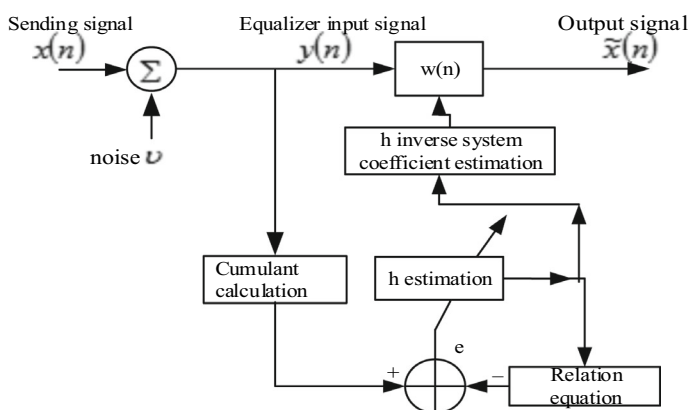
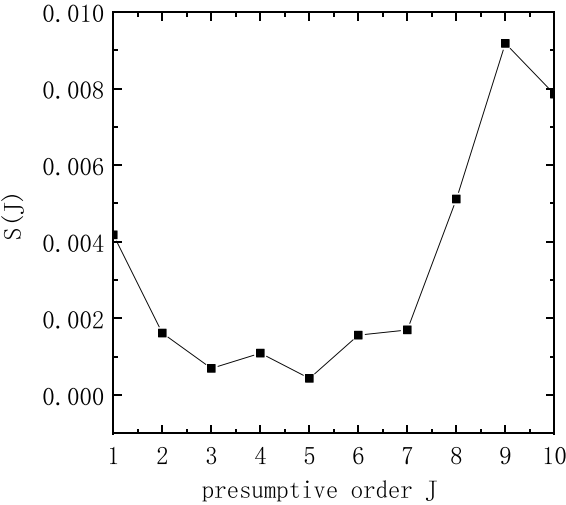


Fig. 9.14 Experimental block diagram 1 of the direct blind-equalization algorithm

Five-hundred independent simulation experiments are carried out with different step lengths, using Eqs. (9.24) and (9.25) to estimate the system order, assuming the MA model order $J = 10$. The estimated value curve of $S(J)$ is obtained as shown in Fig. 9.15, according to Eq. (9.24). The system order L estimated by the direct method is $L = 4$.

The MSE curve of the algorithm simulation is shown in Fig. 9.16. It shows that the convergence speed is faster when *stepsize* is 0.1×10^{-3} ; however, the steady-state

Fig. 9.15 $S(J)$ curve

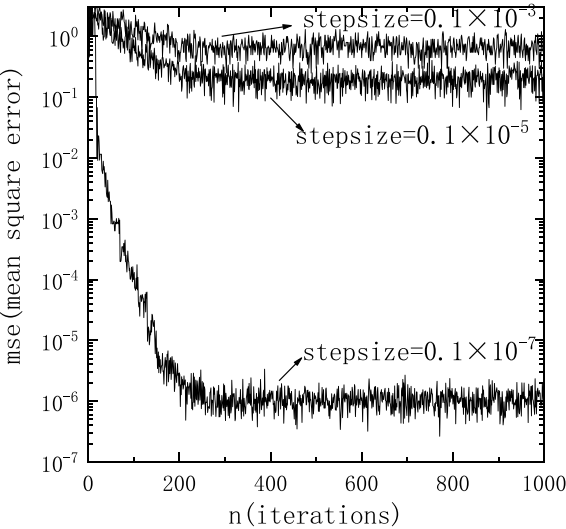


error is larger. When the step size is reduced, the convergence speed and steady-state error gradually decrease.

(2) **Multiplicative noise**

The direct blind-equalization algorithm was applied in a simulation environment with only multiplicative noise, and the block diagram of the algorithm simulation experiment is shown in Fig. 9.17.

Fig. 9.16 Mean squared error (MSE) curve of the algorithm simulation



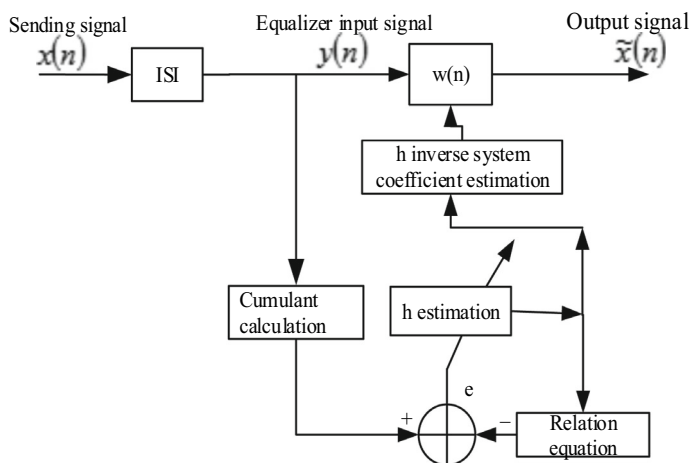


Fig. 9.17 MSE curve of the direct method

The step size was set as $stepsize = 0.1 \times 10^{-7}$. The direct blind-equalization algorithm was independently simulated 500 times in the simulation environment shown in Fig. 9.17. The $S(J)$ curve is shown in Fig. 9.18. The estimated system order L of the algorithm is $L = 2$. The MSE curve obtained from the simulation algorithm is shown in Fig. 9.19. It shows the steady-state error of the direct method algorithm with only multiplicative noise.

Figure 9.20 shows the mean squared error curves comparing the above two simulation environments for the same step length using the direct method. It shows that multiplicative noise has a greater impact on the steady-state error of the algorithm.

Fig. 9.18 $S(J)$ curve 2

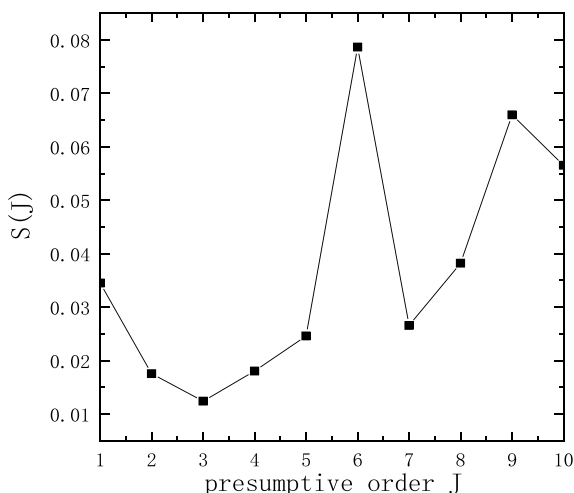


Fig. 9.19 Direct-method
MSE curve

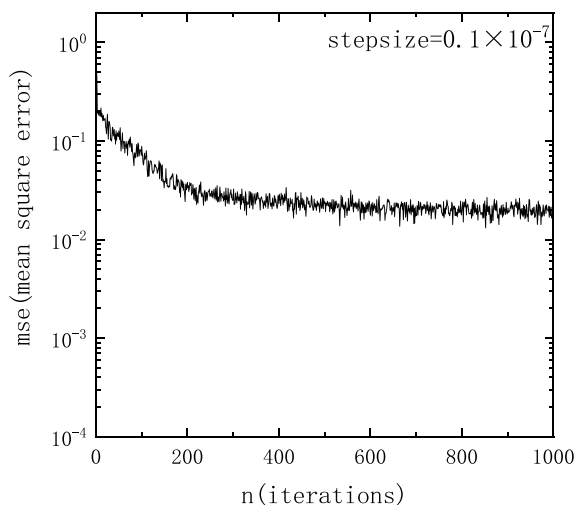
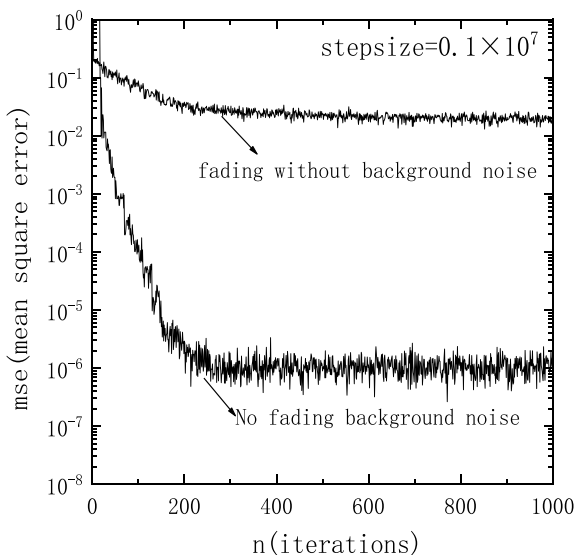


Fig. 9.20 MSE curves of
the normalization method



(3) Both multiplicative and additive noise

The direct blind-equalization algorithm was applied in a simulation environment with multiplicative and additive noise. A block diagram of the algorithm simulation environment is shown in Fig. 9.21, and $stepsize = 0.1 \times 10^{-7}$. Five-hundred independent simulations of the direct blind-equalization algorithm were performed. The $S(J)$ curve is shown in Fig. 9.22; then, the system order L estimated by the algorithm is $L = 2$. The MSE curve of the algorithm is shown in Fig. 9.23.

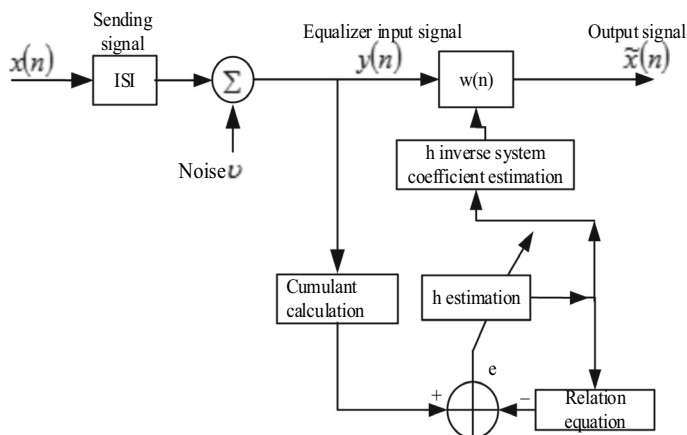
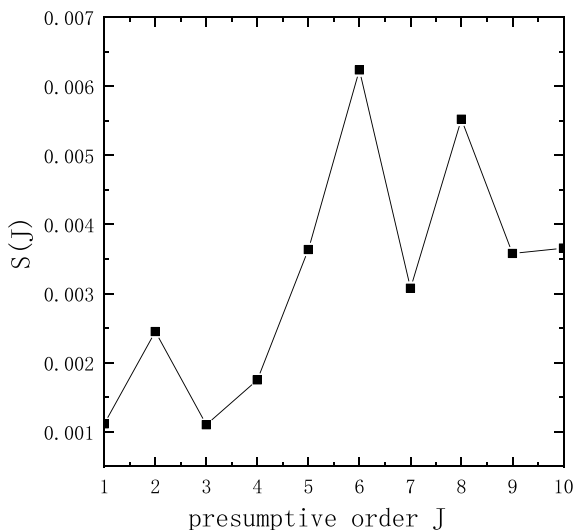


Fig. 9.21 Experimental block diagram 3 of the direct blind-equalization algorithm

Fig. 9.22 $S(J)$ curve



As shown in Fig. 9.23, the convergence speed of the direct method was slower in the case with both multiplicative and additive noise, and the steady-state error was larger. Comparing the performance of the direct method using the above three channel models, the mean squared error curves for a fixed step size is shown in Fig. 9.24. The algorithm converges faster and with less steady-state error with only additive noise, and converges significantly slower and with more steady-state error with multiplicative noise.

Fig. 9.23 MSE curve of the direct method

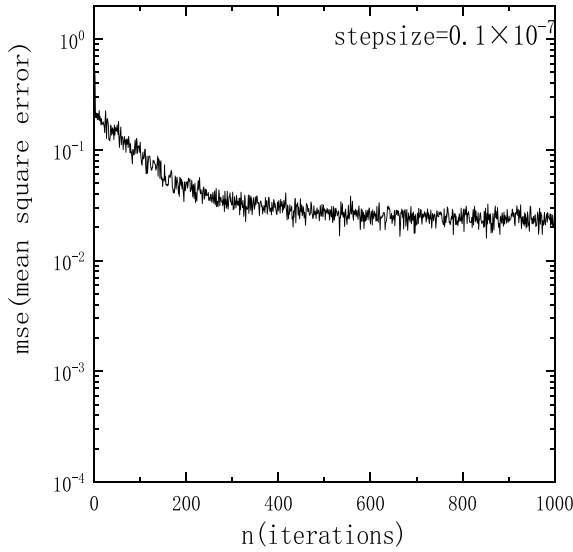
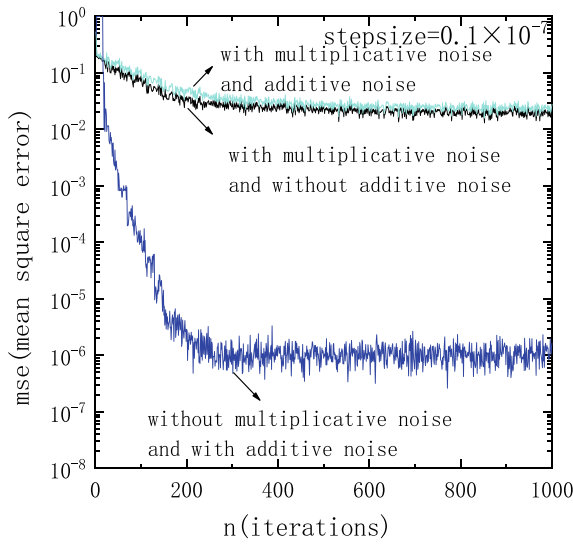


Fig. 9.24 MSE of the direct method



However, the direct method has a faster convergence rate with only multiplicative noise than that with both multiplicative- and additive-noise interference, and the difference in the steady-state error between the algorithms in these two cases is small.

9.2 Analysis of the Bussgang Blind-Equalization Algorithm

Bussgang-like blind-equalization algorithms implicitly use higher-order statistics and an iterative approach to solve for the weight vectors. We discuss the following three classical Bussgang algorithms: the constant-mode algorithm (CMA), the Sato algorithm, and the decision-directed (DD) algorithm. The final focus is on a blind-equalization algorithm combining the DD algorithm with adaptive filtering, based on higher-order cumulants.

9.2.1 Definition of the Bussgang Process

If a stochastic process $x(n)$ satisfies the condition [24]

$$E\{\tilde{x}(n) \cdot \tilde{x}(n+k)\} = E\{g[\tilde{x}(n) \cdot \tilde{x}(n+k)]\}, \quad (9.50)$$

where $g(\bullet)$ is a memoryless nonlinear function, it is called a Bussgang process [1]. The autocorrelation function of a Bussgang process [24] is equal to the mutual correlation between the process and the output of a memoryless nonlinear function using it as a variant. In 1952, Bussgang [25] first discovered that any correlated Gaussian process has the above property. In 1955, Barrett and Lampard [26] proved that all stochastic processes with exponentially decaying autocorrelation functions have this property, extending Bussgang's conclusion [27].

9.2.2 Mathematical Model of Bussgang's Algorithm

We focused on the Bussgang blind-equalization algorithm, based on the baseband channel. Baseband means that modulation, demodulation, coding, decoding, and synchronization are not considered [28]. First, we discuss the equivalent-baseband model of the Bussgang-like blind-equalization algorithm.

The basic principle of the Bussgang blind-equalization algorithm is to first establish a cost function, such that the ideal system corresponds to the minimal value of the cost function, and then use an adaptive algorithm to find the maximum value of the cost function. When the cost function reaches the maximum value, the system becomes ideal. A block diagram of the equivalent-baseband principle for a transmitter-to-receiver system is shown in Fig. 9.25 [24].

In Fig. 9.25, $\{x(n)\}$ denotes the transmit-signal sequence of the transmitter, $h(n)$ is the channel impulse response of length $p + 1$, and $n(n)$ is the channel noise. $\{y(n)\}$ is the input signal to the equalizer. $w(n)$ is the vector of equalizer weight coefficients of length $2L + 1$. $\{\tilde{x}(n)\}$ is the output signal of the equalizer. $e(n)$ is the error signal of the equalizer output, relative to the estimated signal $\{\hat{s}(n)\}$. $g(\bullet)$ is the

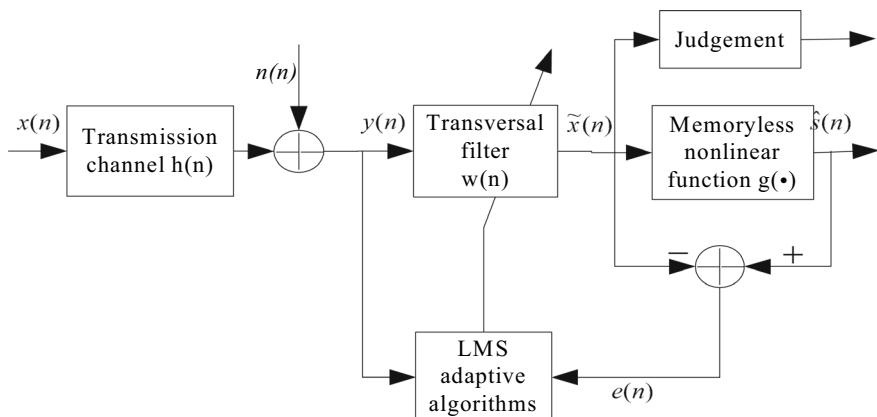


Fig. 9.25 Schematic diagram of a Busgang blind equalizer

nonlinear memoryless estimation function that performs a nonlinear transformation of the equalizer output.

In the study of the blind-equalization algorithm, the channel $h(n)$ can be expressed as $h(n) = [h_0(n), h_1(n), \dots, h_p(n)]^T$, with T denoting a transpose. $w(n)$ is the vector of equalizer weight coefficients, that is, $w(n) = [w_{-L}(n), w_{-L+1}(n), \dots, w_0(n), w_1(n), \dots, w_L(n)]^T$ with length $2L + 1$. The finite impulse-response (FIR) linear equalizer is discussed as an example, and its structure is shown in Fig. 9.26 [18]. The output of the equalizer is

$$\tilde{x}(n) = w^T(n)x(n). \quad (9.51)$$

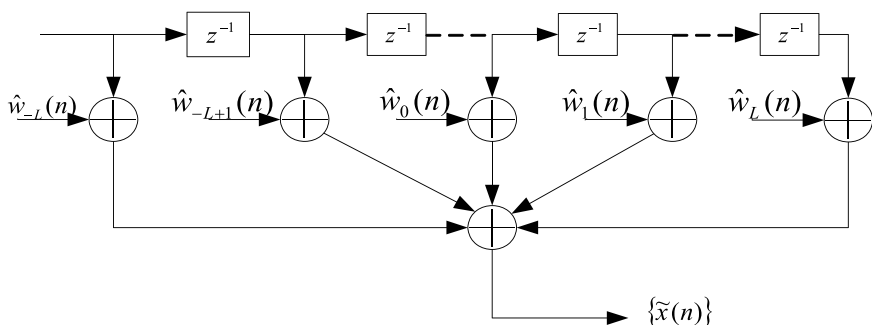


Fig. 9.26 Structure of a FIR linear equalizer

9.2.3 Analysis of the Bussgang Blind-Equalization Algorithm

(1) Sato's algorithm

Sato proposed a blind-equalization algorithm for M -decimal pulse-amplitude modulation (MPAM) signals in 1975 [48]. In the Sato adaptive-equalization algorithm, the minimization cost function is defined as [28]

$$J(n) = E\{[\hat{s}(n) - \tilde{x}(n)]^2\}, \quad (9.52)$$

where $\hat{s}(n)$ is given by the memoryless nonlinear estimator:

$$\hat{s}(n) = \gamma \operatorname{sgn}[\tilde{x}(n)]. \quad (9.53)$$

The constant γ is the gain of the equalizer and is defined as

$$\gamma = \frac{E\{x^2(n)\}}{E\{|x(n)|\}}. \quad (9.54)$$

It can be seen that Sato's algorithm is a special case of the Bussgang algorithm [24]:

$$g(\tilde{x}(n)) = \gamma \operatorname{sgn}[\tilde{x}(n)]. \quad (9.55)$$

(2) Godard's algorithm

Godard first proposed the constant-mode blind-equalization algorithm, which is applicable to the equalization of all transmitted signals with a constant envelope (referred to as constant mode). Its cost function is [29]

$$J_G(n) = E\{|\tilde{x}(n)|^p - R_p\}^2, \quad (9.56)$$

where p is a positive integer, usually set as $p = 1$ or $p = 2$, and R_p is a real constant [29]:

$$R_p = \frac{E\{|x(n)|^{2p}\}}{E\{|x(n)|^p\}}. \quad (9.57)$$

R_p should satisfy the condition that when ideal equalization is obtained, the gradient of cost function $J_G(k)$ is zero. The CMA cost function has been widely studied and applied for $p = 2$.

(3) Decision-directed algorithm

The decision-directed (DD) algorithm was proposed by Lucky in the 1960s. When the Bussgang algorithm converges and the eye diagram is "open," the equalizer

operates in DD mode. The equalizer's transverse filter has the same minimum mean squared error as the tap coefficients, because the minimum mean squared error of the equalizer's lateral filter taps is the same as that of a normal adaptive filter.

In decision-directed mode, the memoryless nonlinear function $g(\bullet)$ only implements the "threshold decision" function. The "threshold decision" means that $g(\bullet)$ judges the output signal $\{\tilde{x}(n)\}$ of the transversal filter in such a manner that the judgment result $\{\hat{s}(n)\}$ is closest to $\{\tilde{x}(n)\}$; that is,

$$\hat{s}(n) = \text{dec}(\tilde{x}(n)), \quad (9.58)$$

where $\text{dec}(\bullet)$ is a judgment function.

For example, for a sequence of binary equal-probability data, if the set of symbols of the emitted signal is $\{-1, +1\}$, only two data cases exist [24]:

$$x(n) = \begin{cases} +1 & \text{for symbol 1} \\ -1 & \text{for symbol 0} \end{cases}. \quad (9.59)$$

The verdict function is [24]

$$\hat{x}(n) = \text{dec}(x(n)) = \text{sgn}(x(n)). \quad (9.60)$$

Then,

$$\hat{x}(n) = \begin{cases} +1 & \text{for } x(n) > 0 \\ -1 & \text{for } x(n) < 0 \end{cases}, \quad (9.61)$$

where $\text{sgn}(\bullet)$ is the sign function. The decision-directed algorithm is the Bussgang algorithm that uses $g(\bullet) = \text{sgn}(\bullet)$.

The three blind-equalization algorithms discussed above, namely Godard's algorithm (of which the CMA algorithm is a special case), Sato's algorithm, and the DD algorithm, are three special cases of Bussgang's algorithm. These three algorithms are typically iterated using the least-mean-square (LMS) criterion with a weight vector iteration of [24]

$$w(n+1) = w(n) + \mu Y^*(n)e(n). \quad (9.62)$$

μ is the step factor and a is the conjugate operation. $e(n)$ is the error function of the algorithm. Several classical stochastic-gradient blind-equalization algorithms are summarized in the literature [28]. Table 9.4 summarizes several stochastic-gradient blind-equalization algorithms mentioned in this paper, where the equalizer error sequence is defined as

$$e(n) = g(\tilde{x}(n)) - \tilde{x}(n). \quad (9.63)$$

Table 9.4 Error functions of three Bussgang blind-equalization algorithms

DDLMS	$e_{DD} = \tilde{s}(n) - \tilde{x}(n)$
Sato	$e(n) = \gamma \cdot \text{sgn}[\tilde{x}(n)] - \tilde{x}(n) \quad \gamma = \frac{E\{x^2(n)\}}{E\{ x(n) \}}$
Godard	$e(n) = \tilde{x}(n)(R_p - \tilde{x}(n) ^p) \quad R_p = \frac{E\{ x(n) ^{2p}\}}{E\{ x(n) ^p\}}$

(4) DD Bussgang/CDLMS blind-equalization algorithm

Although the blind convergence of the DD algorithm is weak, its steady-state tracking performance is good. There are two broad methods to make the DD blind-equalization algorithm have both a fast convergence rate and small residual error. The first method is to enhance the DDLMS algorithm by improving its blind-convergence capability; a typical example is the stop-and-go algorithm.

The second method is to combine the DDLMS algorithm with another blind-equalization algorithm (which has a stronger blind-convergence capability, but poorer steady-state performance), so that the hybrid algorithm has both a strong blind-convergence capability and maintains the DDLMS algorithm's good steady-state performance.

In reference [30], a switching strategy between the Sato algorithm and the DDLMS algorithm is proposed; in reference [31], a strategy that switches between the CMA algorithm and the DDLMS algorithm is proposed. In references [32, 33], another switching strategy between the CMA algorithm and the DDLMS algorithm is proposed.

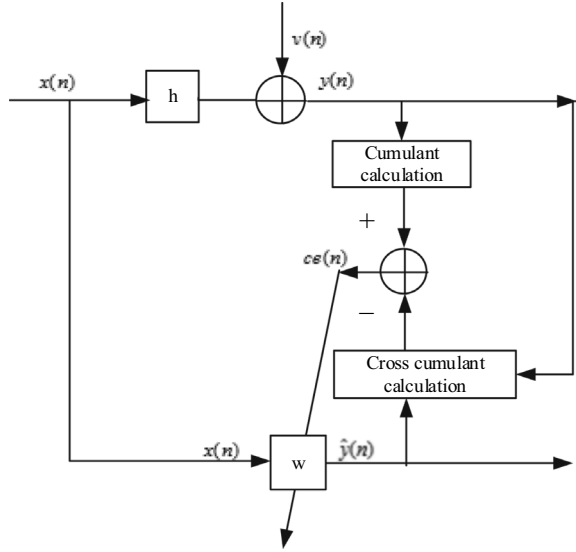
First, the principle of cumulant-based adaptive filtering is examined. Figure 9.27 shows a block diagram of the principle of cumulant-based adaptive filtering. The cost functions used by typical LMS adaptive-filtering algorithms are all second-order statistics, whereas the cumulant-based LMS adaptive-filtering algorithm (CDLMS) utilizes higher-order cumulants as the cost function.

In Fig. 9.27, $\{x(n)\}$ is a real, zero-mean, smooth, independent, identically distributed non-Gaussian sequence. $\{v(n)\}$ is independent of zero-mean white or colored Gaussian noise. $H^T = [h(0), h(1), \dots, h(p)]$ represents the unit response of an invariant linear system. $W^T = [w_0, w_1, \dots, w_p]$ is the weighted coefficient vector of a $p + 1$ first-order finite impulse-response (FIR) filter. $\{y(n)\}$ is the desired output of the filter; that is, $y(n) = \sum_{i=0}^p h(i)x(n) + v(n)$. $\{\hat{y}(n)\}$ is the actual output of

the filter; that is, $y(n) = \sum_{i=0}^p w(i)x(n)$. $ce(n)$ represents the error function based on the cumulative value:

$$\begin{aligned}
 ce(\tau_1, \tau_2, \dots, \tau_{k-1}; n) &= c_{yy\dots y}(\tau_1, \tau_2, \dots, \tau_{k-1}; n) - c_{\hat{y}y\dots y}(\tau_1, \tau_2, \dots, \tau_{k-1}; n) \\
 &= c_{yy\dots y}(\tau_1, \tau_2, \dots, \tau_{1-k}; n) \\
 &\quad - \sum_{i=0}^p w_i c_{xy\dots y}(\tau_1 + i, \tau_2 + i, \dots, \tau_{k-1} + i; n). \quad (9.64)
 \end{aligned}$$

Fig. 9.27 Schematic block diagram of LMS adaptive filtering (CDLMS), based on high-order cumulants [34]



The cumulant-based error criterion $J(n)$ [35] is

$$\begin{aligned}
 J(n) &= \sum_{t=1}^n \sum_{\tau_1} \dots \sum_{\tau_{k-1}} \beta^{n-1} ce^2(t) \\
 &= \sum_{t=1}^n \beta^{n-1} (C_{yy\dots y}(t) - C_{xy\dots y}(t)W)^T (C_{yy\dots y}(t) - C_{xy\dots y}(t)W), \quad (9.65)
 \end{aligned}$$

where β is the forgetting factor, $0 < \beta < 1$; $ce(t)$ is the cumulative error; $(\tau_1, \tau_2, \dots, \tau_{k-1})$ is the slip M , that is, the number of points in the priority domain; and $C_{xy\dots y}(t)$ is the matrix of mutual accumulations of size $M * (p + 1)$ with the t th row:

$$[c_{xy\dots y}(\tau_1, \tau_2, \dots, \tau_{k-1}; t), \dots, c_{xy\dots y}(\tau_1 + p, \tau_2 + p, \dots, \tau_{k-1} + p; t)]. \quad (9.66)$$

$C_{yy\dots y}(t)$ is a column vector of size $M * 1$ representing the self-cumulative quantity of the desired output, whose t th row is

$$c_{yy\dots y}(\tau_1, \tau_2, \dots, \tau_{k-1}; t). \quad (9.67)$$

The recursive formula for updating the weight vector of the CDLMS adaptive filter is [35]

$$\bar{w}(n+1) = \bar{w}(n) - \mu(n)g(n). \quad (9.68)$$

In Eq. (9.68), $g(n)$ is the gradient of the CDLMS algorithm [35]:

$$\begin{aligned} g(n) &= 2 \sum_{i=0}^n \beta^{n-i} [c_{xy\dots y}^T(n) c_{xy\dots y}(n) \bar{w}(n) - c_{xy\dots y}^T(n) c_{yy\dots y}(n)] \\ &= 2(\varphi(n) \bar{w}(n) - \theta(n)), \end{aligned} \quad (9.69)$$

where

$$\theta(n) = \sum_{i=0}^n \beta^{n-i} c_{xy\dots y}^T(n) c_{yy\dots y}(n) = \beta \theta(n-1) + c_{xy\dots y}^T(n) c_{yy\dots y}(n) \quad (9.70)$$

$$\varphi(n) = \sum_{i=0}^n \beta^{n-i} c_{xy\dots y}^T(n) c_{xy\dots y}(n) = \beta \varphi(n-1) + c_{xy\dots y}^T(n) c_{xy\dots y}(n). \quad (9.71)$$

For the decision-directed (DD) Bussgang blind-equalization algorithm, the tapped coefficients $w(n)$ of the blind equalizer are updated using the higher-order cumulant-based LMS steepest-descent adaptive-filtering algorithm, which is theoretically effective in suppressing not only additive noise but also symmetrically distributed non-Gaussian colored noise [18].

A schematic block diagram of the blind-equalization algorithm combining the DD Bussgang algorithm with the CDLMS adaptive-filtering algorithm is shown in Fig. 9.28.

In Fig. 9.28, $\{\tilde{x}(n)\}$ is the output of the equalizer. $\{\hat{s}(n)\}$ is the estimated signal from the equalizer. $\{\hat{s}(n)\}$'s k th-order cumulant is defined as

$$c_{\hat{s}\hat{s}\dots\hat{s}}(\tau_1, \tau_2, \dots, \tau_{k-1}; n) = \text{cum}(\hat{s}(n), \hat{s}(n + \tau_1), \dots, \hat{s}(n + \tau_{k-1})). \quad (9.72)$$

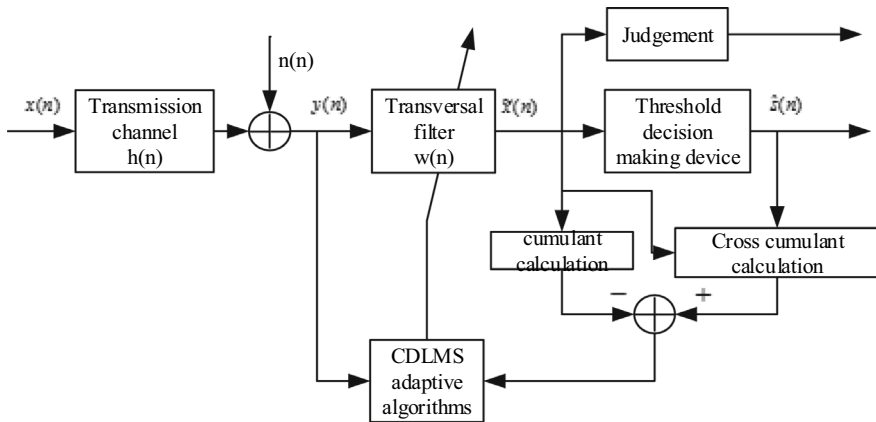


Fig. 9.28 Block diagram of the DD Bussgang/CDLMS blind-equalization algorithm

The k th-order mutual accumulation of $\{\tilde{x}(n)\}$ and $\{\hat{s}(n)\}$ is

$$\begin{aligned}
 c_{\tilde{x}\hat{s}\hat{s}}(\tau_1, \tau_2, \dots, \tau_{k-1}; n) &= cum(\tilde{x}(n), \hat{s}(n + \tau_1), \dots, \hat{s}(n + \tau_{k-1})) \\
 &= \sum_{i=0}^p w_i cum(y(n), \hat{s}(n + \tau_1 + i), \dots, \hat{s}(n + \tau_{k-1} + i)) \\
 &= \sum_{i=0}^p w_i c_{y\hat{s}\dots\hat{s}}(\tau_1 + i, \dots, \tau_{k-1} + i; n).
 \end{aligned} \tag{9.73}$$

For an adaptive filter, the cost function is the mean squared error between the equalizer output and the desired signal; however, for blind-equalization, the desired signal is unknown. Therefore, an estimate of the equalizer output signal is used to approximate the desired channel, and the cost function of the blind equalizer can be expressed as follows:

$$\begin{aligned}
 J(n) &= \sum_{t=1}^n \sum_{\tau_1} \dots \sum_{\tau_{k-1}} \beta^{n-1} ce^2(t) \\
 &= \sum_{t=1}^n \beta^{n-1} (C_{\tilde{x}\tilde{x}\dots\tilde{x}}(t) - C_{\hat{s}\tilde{x}\dots\tilde{x}}(t)W)^T (C_{\tilde{x}\tilde{x}\dots\tilde{x}}(t) - C_{\hat{s}\tilde{x}\dots\tilde{x}}(t)W).
 \end{aligned} \tag{9.74}$$

The error function $ce(n)$, based on the cumulative amount, is expressed as

$$\begin{aligned}
 ce(\tau_1, \tau_2, \dots, \tau_{k-1}; n) &= c_{\tilde{x}\tilde{x}\dots\tilde{x}}(\tau_1, \tau_2, \dots, \tau_{k-1}; n) - \sum_{i=0}^p w_i c_{\hat{s}\tilde{x}\dots\tilde{x}}(\tau_1 + i, \tau_2 + i, \dots, \tau_{k-1} + i; n).
 \end{aligned} \tag{9.75}$$

DD Bussgang/CDLMS Blind-Equalization Algorithm Simulation

In the following, the DD Bussgang algorithm is combined with the third-order cumulant-based CDLMS adaptive-filtering algorithm, for three typical channel models. We chose the third-order cumulant-based cost function, which can be expressed as [36]

$$\begin{aligned}
 J(n) &= \sum_{t=1}^n \sum_{\tau_1} \sum_{\tau_2} \beta^{n-1} ce^2(t) \\
 &= \sum_{t=1}^n \beta^{n-1} (C_{\tilde{x}\tilde{x}\tilde{x}}(t) - C_{\hat{s}\tilde{x}\tilde{x}}(t)W)^T (C_{\tilde{x}\tilde{x}\tilde{x}}(t) - C_{\hat{s}\tilde{x}\tilde{x}}(t)W).
 \end{aligned} \tag{9.76}$$

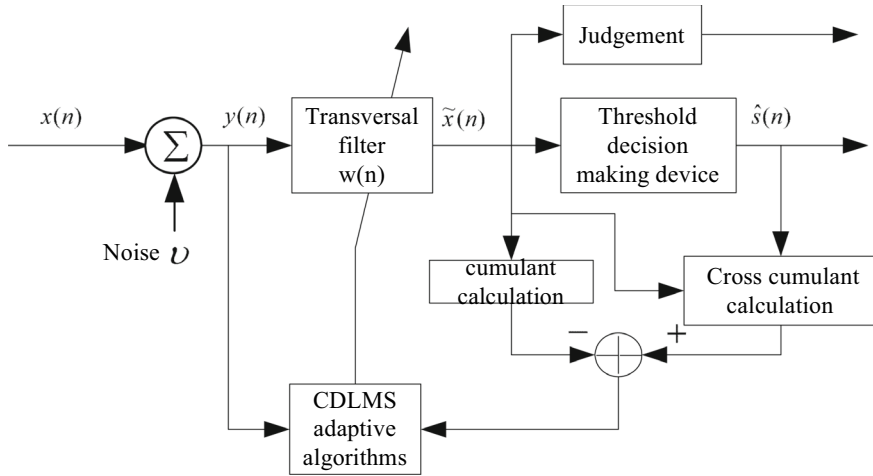


Fig. 9.29 Experimental block diagram of the DD Bussgang/CDLMS blind-equalization algorithm simulation (I)

(1) Additive noise

A blind-equalization algorithm combining the DD Bussgang algorithm with the CDLMS algorithm was applied in a simulation environment with only additive noise. The block diagram of the simulation experiment is shown in Fig. 9.29.

Five-hundred independent simulations were conducted with different step sizes. The mean squared error (MSE) curves were obtained, as shown in Fig. 9.30.

As can be seen from Fig. 9.30, the convergence speed is faster when *stepsize* is 0.1; however, the steady-state error is larger. When *stepsize* is reduced, the convergence speed gradually decreases, along with the steady-state error.

(2) Multiplicative noise

In a simulation environment with only multiplicative noise, the blind-equalization algorithm combining the DD Bussgang algorithm and the CDLMS algorithm was applied. The block diagram of the algorithm simulation experiment is shown in Fig. 9.31.

The step size was taken as $stepsize = 0.1 \times 10^{-3}$, and the DD Bussgang/CDLMS blind-equalization algorithm was simulated 500 times independently to obtain the MSE curve shown in Fig. 9.32.

Figure 9.32 shows that in the case with only multiplicative noise, the steady-state error of the blind-equalization algorithm under study is larger, relative to the case with only additive noise.

(3) Multiplicative and additive noise

The direct blind-equalization algorithm was applied in a simulation environment with multiplicative and additive noise. The block diagram of the algorithm simulation experiment is shown in Fig. 9.33.

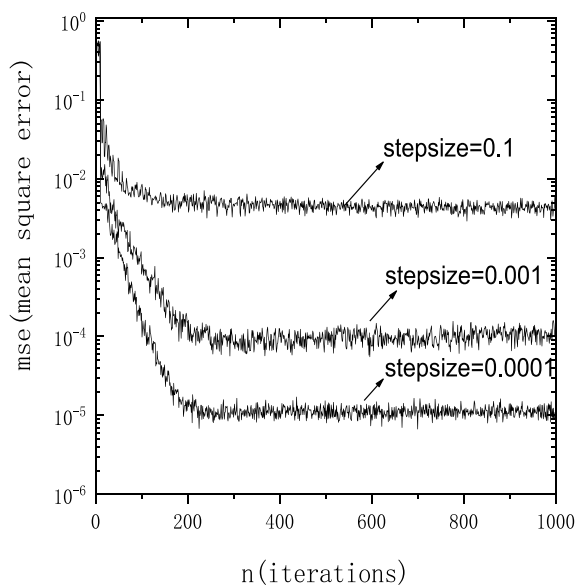


Fig. 9.30 MSE curve of DD Bussgang/CDLMS blind equalization algorithm

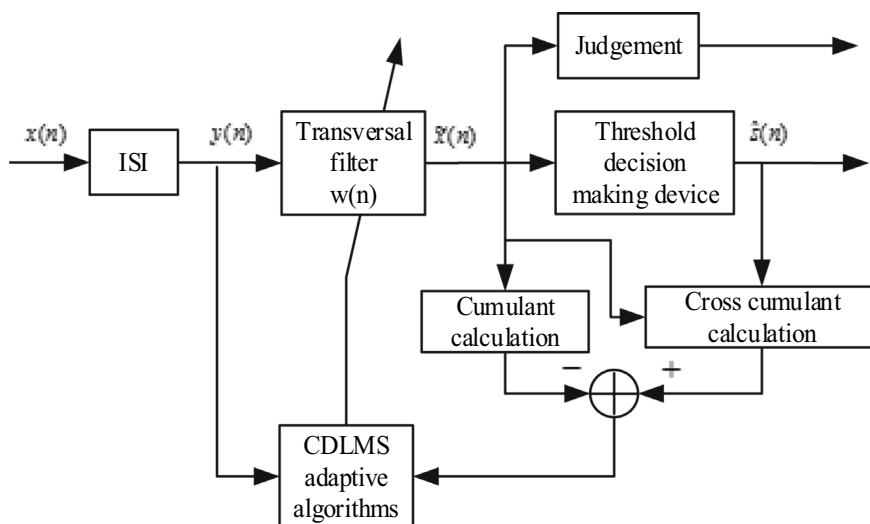


Fig. 9.31 Block diagram of the DD Bussgang/CDLMS blind-equalization algorithm simulation experiment (II)

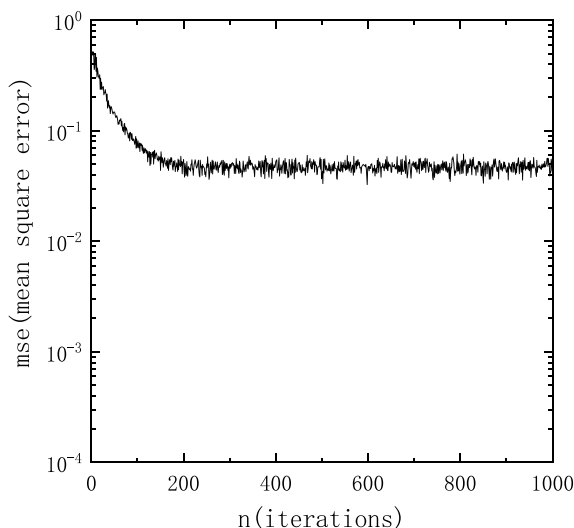


Fig. 9.32 Mean square error (MSE) curve of DD Bussgang/CDLMS blind equalization algorithm with multiplicative noise

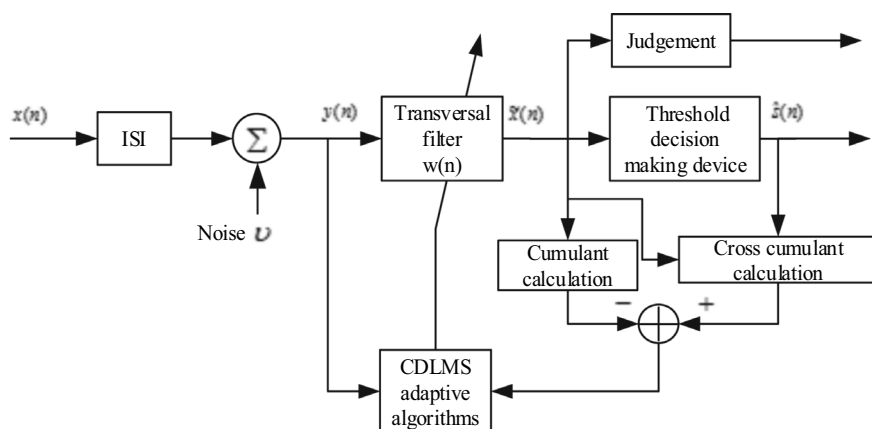
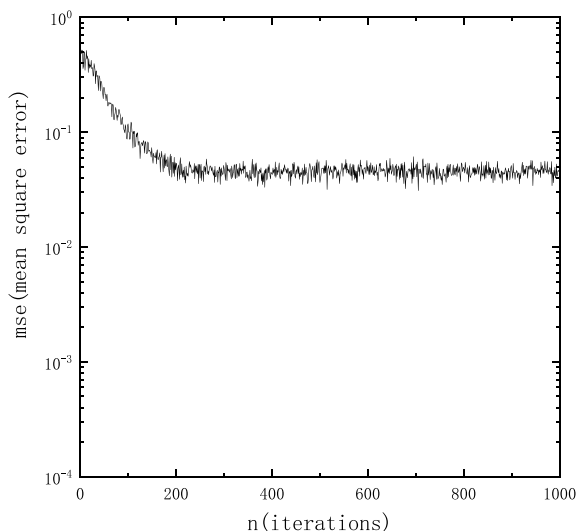


Fig. 9.33 Experimental block diagram of the DD Bussgang/CDLMS blind-equalization algorithm simulation (III)

The step size $stepsize = 0.1 \times 10^{-3}$. Five-hundred independent simulations were conducted for the DD Bussgang/CDLMS blind-equalization algorithm, and the MSE curve was obtained, as shown in Fig. 9.34. It can be seen that the convergence speed of this blind-equalization algorithm with multiplicative and additive noise is slower, and the steady-state error is larger.

Fig. 9.34 MSE curve of DD Bussgang/CDLMS blind equalization algorithm



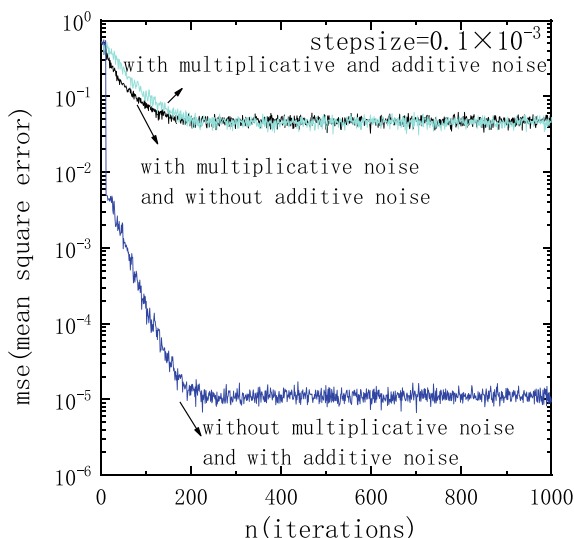
We compare the performance of the DD Bussgang/CDLMS blind-equalization algorithm, based on third-order cumulants, in simulation environments based on the above three channel models. We present the mean squared error curve with the same step length, as shown in Fig. 9.34. In the case of only additive noise, the convergence speed of the algorithm is fast and the steady-state error is small; in the case of additive noise, the convergence speed is obviously slower and the steady-state error is relatively increased.

However, similar to the case of additive noise, the blind-equalization algorithm under study has a faster convergence speed in the case with no additive noise than in the case of additive noise. The difference in steady-state error between the algorithms in these two cases is small.

The simulation results above show that the convergence performance of the DD Bussgang/CDLMS blind-equalization algorithm is improved, compared to the DD Bussgang blind-equalization algorithm, as evidenced by a reduction in the steady-state error.

Figure 9.35 shows the mean squared error curves for the DD Bussgang algorithm combined with the CDLMS adaptive-filtering algorithm and the DD Bussgang method, respectively, when using the optimal step size for their algorithms. As can be seen in Fig. 9.35, the steady-state error in the DD Bussgang algorithm combined with the CDLMS adaptive-filtering algorithm is improved over the classical DD Bussgang algorithm, although the convergence rate is slower than that of the classical DD Bussgang algorithm.

Fig. 9.35 Comparison of DD Busgang/CDLMS blind equalization algorithm and DD Busgang blind equalization algorithm

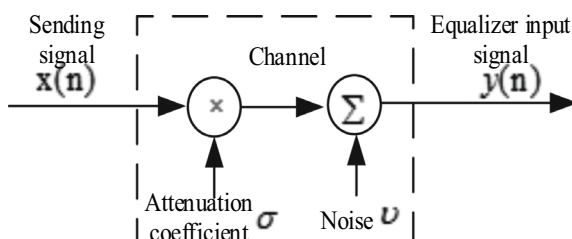


9.3 Channel Equalization in Optical Wireless Communication

We detected the power data at the receiving end, based on actual measurements in foggy, cloudy, light rain, moderate rain, and heavy rain environments with a constant transmit power of 10 mW [37]. We normalized these received power data to obtain the attenuation coefficient σ of the channel to the transmit signal, plus additive Gaussian white noise v with a mean of 0 and variance of 0.01, and used it as the channel model of the atmospheric laser-communication system. The atmospheric laser-communication channel model [38] is represented in Fig. 9.36, where a randomly generated binary sequence is used for the transmit signal $\{x(n)\}$, which passes through the simulated atmospheric laser-communication channel to obtain the equalizer in/out signal $\{y(n)\}$.

We applied this signal to a simulated atmospheric laser channel model to observe the performance of several typical algorithms:

Fig. 9.36 Atmospheric laser-communication channel model



1. DD Bussgang/CDLMS blind-equalization algorithm,
2. Normalization method based on higher-order cumulants, and
3. Direct method based on higher-order cumulants.

9.3.1 Simulation of the Combined DD Bussgang/CDLMS Algorithm for Blind Equalisation

We simulated the decision-directed (DDLMS) method combined with the third-order cumulant-based CDLMS blind-equalization algorithm in atmospheric laser-communication channels under different weather conditions, including heavy rain, moderate rain, light rain, foggy days, and cloudy days. The block diagram of the simulation experiment is shown in Fig. 9.37.

In Fig. 9.37, the initial weight vector of the equalizer is taken as $w(0) = (0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0)$, $stepsize$ is taken as 0.1×10^{-3} , and the atmospheric laser channel model under different weather conditions is simulated for 500 independent simulations. The following set of mean squared error curves is obtained, as shown in Figs. 9.38a–e.

From the above results, it can be observed that the decision-directed (DDLMS) method combined with the blind-equalization algorithm, based on a higher-order cumulative adaptive-filtering algorithm, results in faster algorithm convergence and smaller steady-state errors under various weather conditions. However, the algorithm is computationally intensive and it is more difficult to perform real-time data processing.

(3, 2)-Order Normalized Cumulant Algorithm

We applied the (3, 2)-order normalized cumulative blind-equalization algorithm to atmospheric laser-communication channels under different weather conditions,

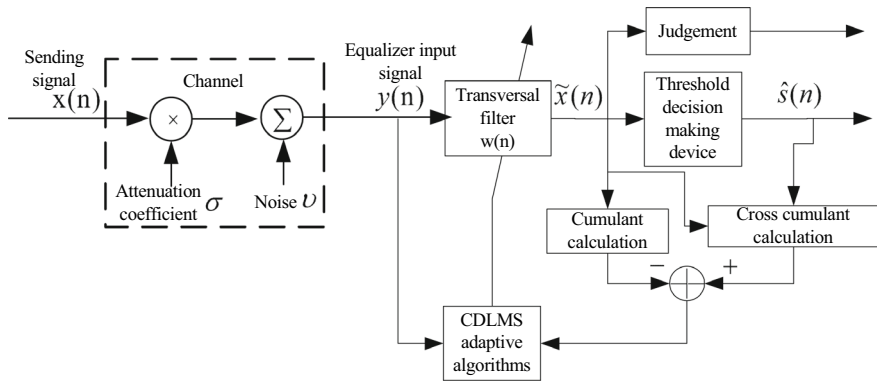


Fig. 9.37 Experimental block diagram of the DD Bussgang/CDLMS blind-equalization algorithm [39]

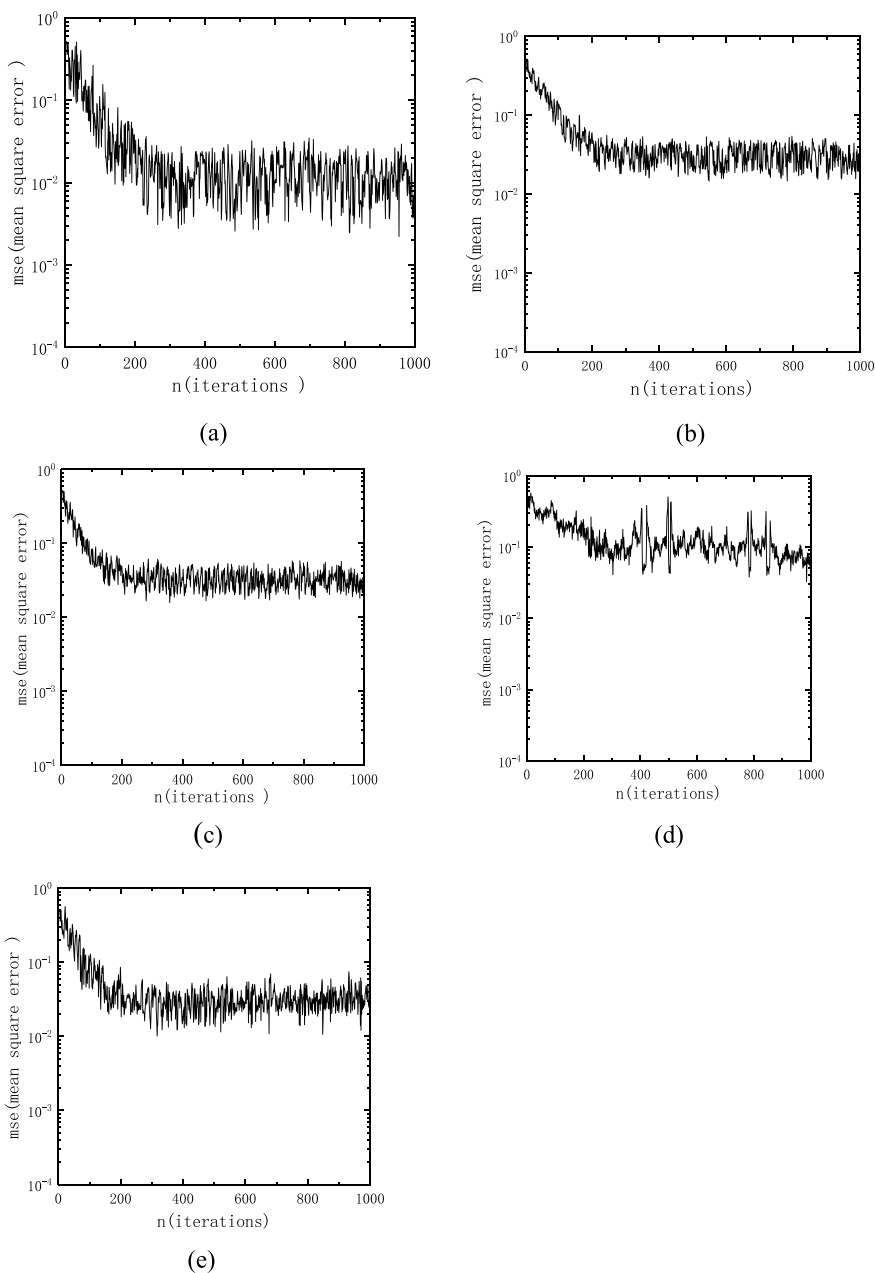


Fig. 9.38 MSE curves of the DD Bussgang/CDLMS algorithm simulated in different weather conditions [39]. **a** Heavy rain [39], **b** Moderate rain [39], **c** Light rain [39], **d** Cloudy weather [39], **e** Foggy weather [39]

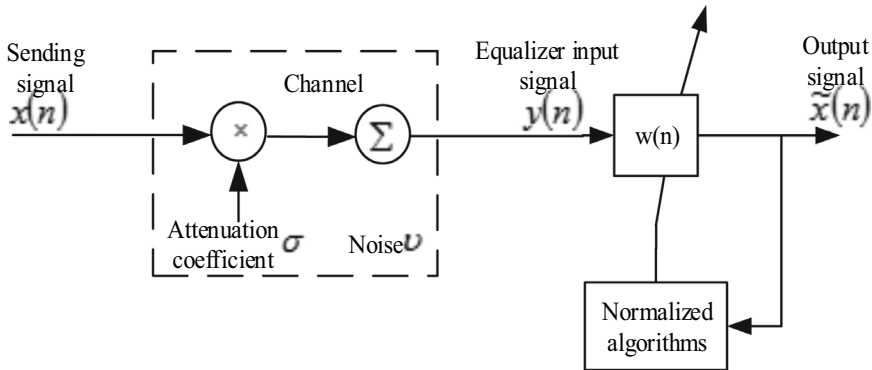


Fig. 9.39 Experimental block diagram of the blind-equalization algorithm, based on third- and second-order normalized cumulants [39]

including heavy rain, moderate rain, light rain, foggy days, and cloudy days. The block diagram of the simulation experiment is shown in Fig. 9.39.

The (3, 2)-order normalized cumulative blind-equalization algorithm was simulated with $stepsize = 0.1 \times 10^{-7}$. Five-hundred independent simulations were conducted under different weather conditions to obtain the following set of MSE curves, as shown in Fig. 9.40a–e.

From the above results, it can be seen that the (3, 2)-order normalized cumulant blind-equalization algorithm has a better equalization effect under different weather conditions and converges faster. Moreover, the algorithm is less computationally intensive and suitable for real-time data processing.

9.3.2 Direct Method

We simulated the third-order cumulant-based direct blind-equalization algorithm in a simulated atmospheric laser-communication channel under different weather conditions. The block diagram of the simulation experiment is shown in Fig. 9.41.

$stepsize$ was set to 0.1×10^{-7} . Five-hundred independent simulations were performed under different weather conditions, and the estimated $S(J)$ curves were obtained, as shown in Figs. 9.42a–e.

Based on the set of $S(J)$ curves represented in Fig. 9.42a–e, the direct method estimates of the system order for each weather condition can be derived according to Eq. (2.92), as shown in Table 9.5.

The following set of MSE curves obtained from the direct-method simulations under different weather conditions are shown in Figs. 9.43a–e.

From the above results, it can be seen that the direct method based on higher-order cumulants has a faster algorithm convergence and smaller steady-state error under various weather conditions; however, its computational size lies between the others.

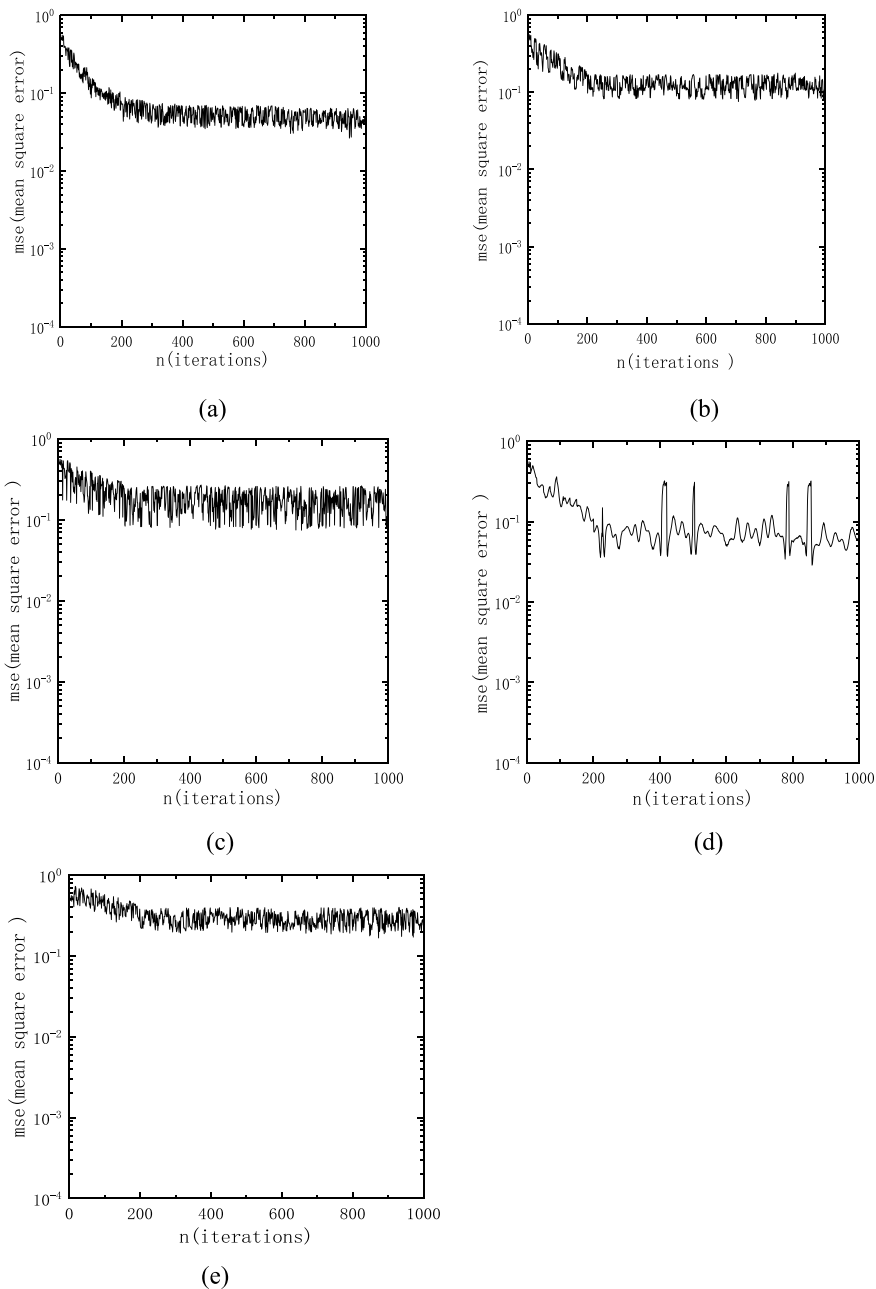


Fig. 9.40 MSE curves under different weather conditions, based on the third- and second-order normalized cumulant blind-equalization algorithm [39]. **a** Heavy rain [39], **b** Moderate rain [39], **c** Light rain [39], **d** Cloudy weather [39], **e** Foggy weather [39]

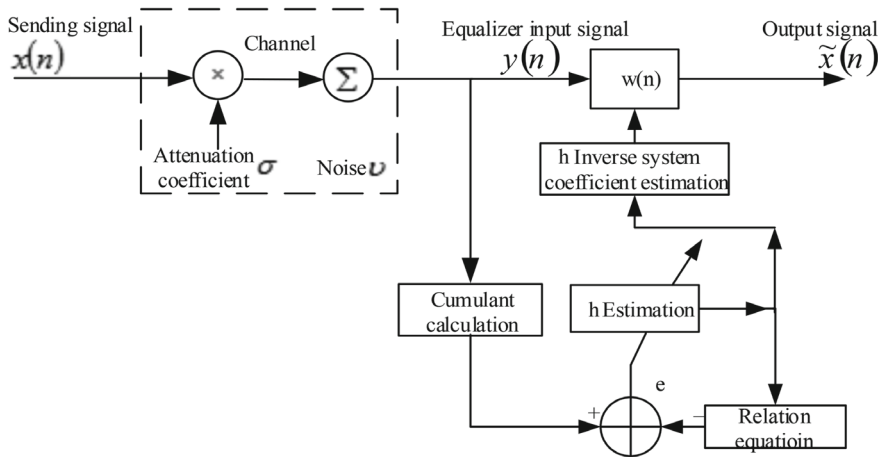


Fig. 9.41 Experimental block diagram of the direct blind-equalization algorithm based on third-order cumulants [39]

Among the three algorithms, the direct method has the fastest convergence speed and best algorithm stability, followed by the (3, 2)-order normalized cumulant blind-equalization algorithm, and finally, the direct-decision (DDLMS) method combined with the CDLMS adaptive-filtering algorithm.

9.3.3 Step Size and Parameter Selection

Optical wireless communication systems are subject to different weather conditions. The actual measurement data vary, and the step size and parameters can vary for the same blind-equalization algorithm. For the adaptive LMS algorithm, there is a clear relationship between the step-size factor and the convergence of the algorithm; that is, the step-size factor should satisfy the following equation [13]:

$$0 < \mu < \frac{2}{\lambda_{\max}}, \quad (9.77)$$

where λ_{\max} is the maximum eigenvalue of the correlation matrix. It is difficult to determine such a clear relationship equation for the blind-equalization algorithm. In general, the larger the step factor, the faster the algorithm converges; however, the equalization effect is not very good. Conversely, the algorithm converges more slowly, but the equalization is better.

We created different settings for the step sizes of the algorithms for different weather conditions. Table 9.6 lists the step-size choices for several blind-equalization algorithms. The normalization forgetting factor is set differently for the different weather conditions, as shown in Table 9.7. As seen in Table 9.7, the forgetting-factor

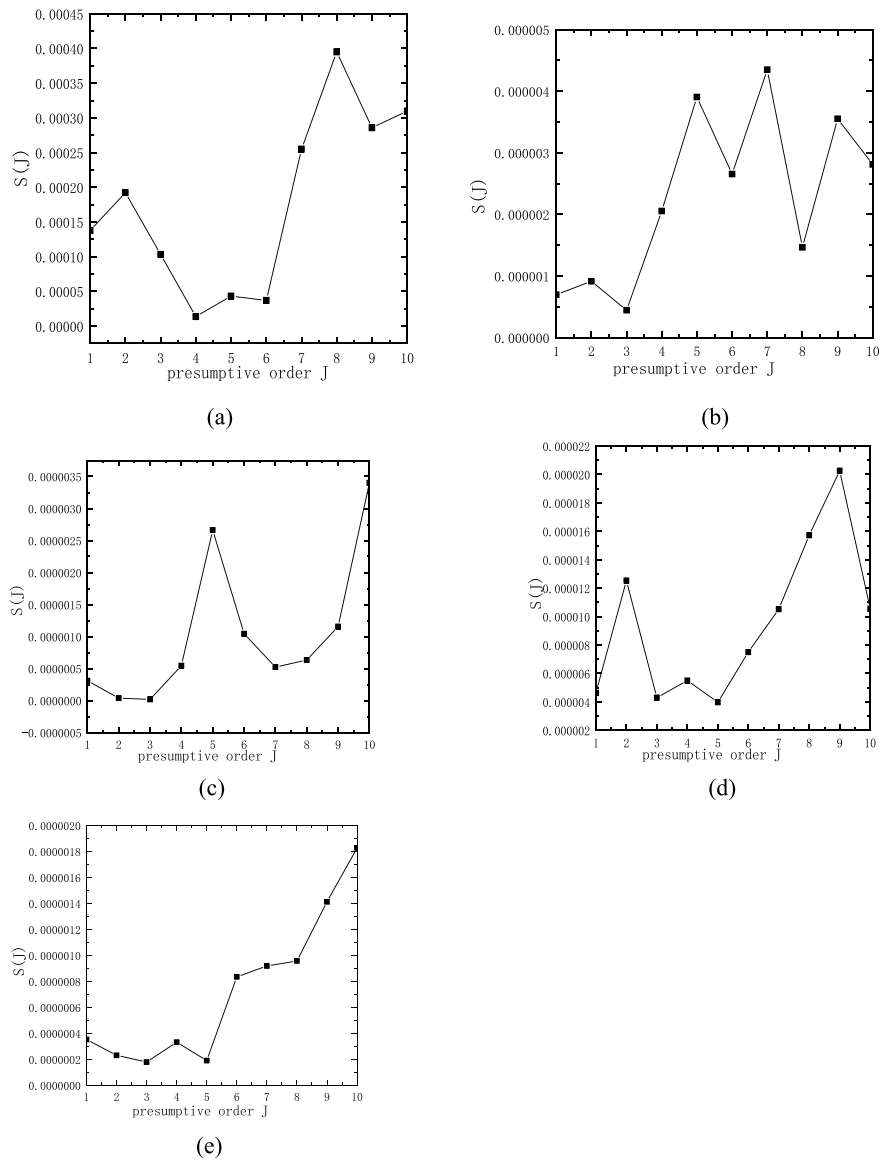


Fig. 9.42 $S(J)$ curves of the direct blind-equalization algorithm based on third-order cumulants under different weather conditions [39], **a** Heavy rain [39], **b** Moderate rain [39], **c** Light rain [39], **d** Cloudy weather [39], **e** Foggy weather [39]

Table 9.5 System-order estimation by the direct method under different weather conditions [39]				
Heavy	Moderate rain	Light rain	Cloud	Fog
3	2	2	4	2

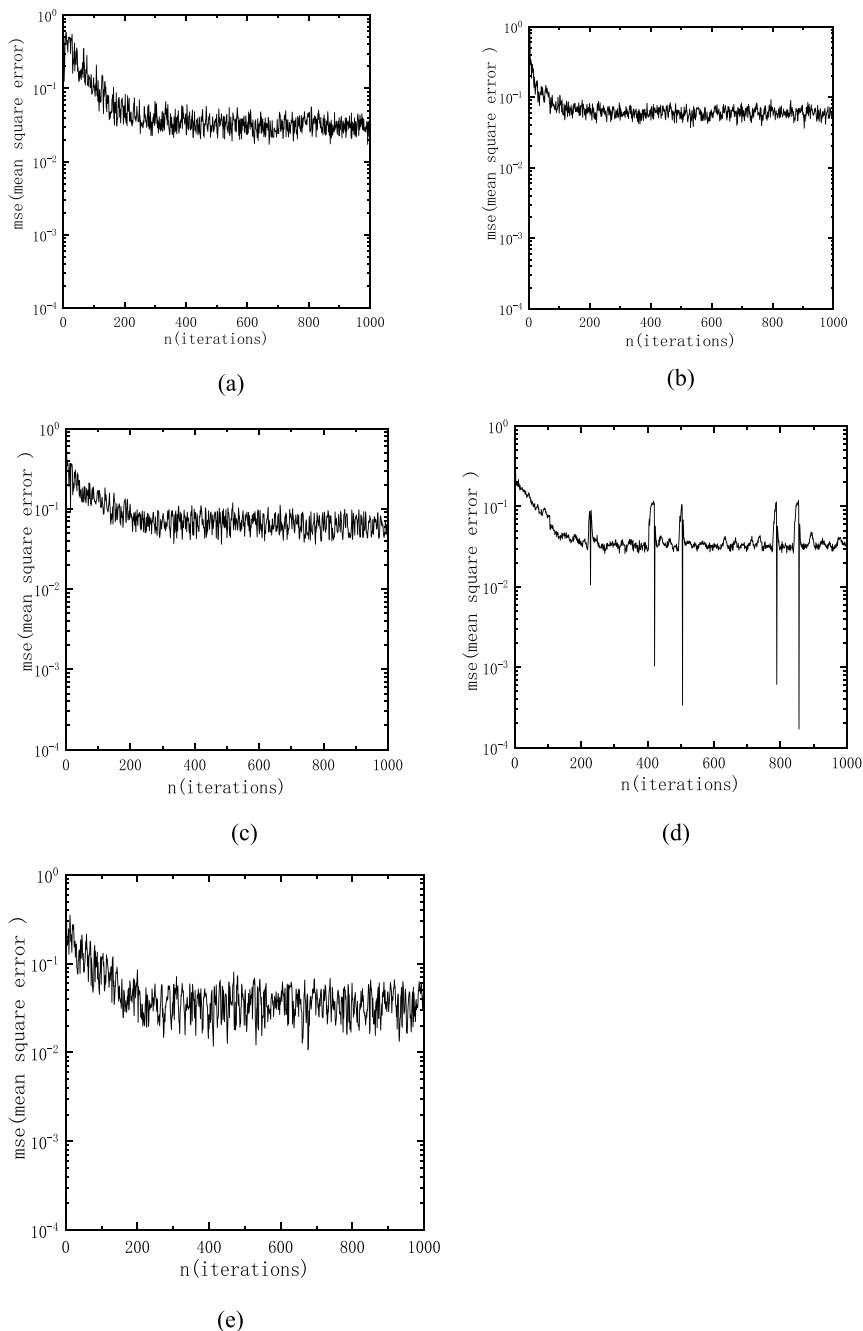


Fig. 9.43 MSE curves of the direct blind-equalization algorithm under different weather conditions [39]. **a** Heavy rain [39], **b** Moderate rain [39], **c** Light rain [39], **d** Cloudy weather [39], **e** Foggy weather [39]

Table 9.6 Step-size selection of several blind-equalization algorithms in the same weather [39]

Algorithms	Weather				
	Heavy rain	Moderate rain	Light rain	Fog	Cloud
CDLMS + DD blind equalization algorithms	≤ 0.5	≤ 0.5	≤ 0.5	≤ 0.5	≤ 0.6
Third and second order normalization method	$\leq 0.1 \times 10^{-3}$	$\leq 0.3 \times 10^{-4}$	$\leq 0.7 \times 10^{-3}$	$\leq 0.4 \times 10^4$	$\leq 0.1 \times 10^{-3}$
Direct method	$\leq 0.1 \times 10^{-2}$	$\leq 0.1 \times 10^{-3}$	$\leq 0.1 \times 10^{-2}$	$\leq 0.1 \times 10^{-2}$	$\leq 0.1 \times 10^{-3}$

Table 9.7 Parameter settings of the normalized cumulant algorithm based on order (3,2) [39]

Weather	Parameter			
	ρ	δ	α	β
Heavy rain	0.9966	0.9966	0.9968	0.9966
Moderate rain	0.9966	0.996	0.995	0.995
Light rain	0.9982	0.997	0.9965	0.993
Fog	0.9985	0.997	0.9974	0.9966
Cloud	0.9982	0.997	0.995	0.995

changes are quite small; however, they have a great impact on the normalization-algorithm performance. Small changes can change the results; thus, the normalized blind-equalization algorithm has high requirements for parameter settings.

References

1. Sato Y (1975) A method of self-recovering equalization for multilevel amplitude-modulation systems. *IEEE Trans [legacy, pre-1988]* 23(6):679–682
2. Godard D (1980) Self-recovering equalization and carrier tracking in two-dimensional data communication systems. *IEEE Trans [legacy,pre-1988]*, 1980, 28(11):1867–1875
3. Benveniste A, Goursat M, Ruget G (1980) Robust identification of a nonminimum phase system: blind adjustment of a linear equalizer in data communications. *IEEE Trans Autom Control* 25:385–399
4. Zhang L, Zhang X (2003) Analysis of blind equalization algorithm based on Bussgang technology. *Comput Engin Appl* 35:74–76
5. Halzinakos D, Nikias CL (1989) Blind decision feedback equalization structures based on adaptive cumulant techniques. In: *ICC’89* 1989:1278–1282
6. Li X (2005) Blind channel estimation and equalization in wireless sensor networks based on correlations among sensors. *IEEE Trans Signal Process* 53(4):1511–1519
7. Wang N (2006) Research on blind equalization in data communication system. Jilin: Jilin University, pp 8–12

8. Benveniste A, Goutsat M, Ruget G (1980) Analysis of stochastic approximation schemes with discontinuous and dependent forcing terms with applications to data communication algorithms. *IEEE Trans AC* 25:1042–1058
9. Hatzinakos D, Nikias CL (1991) Blind equalization using a tricepstrum-based algorithm. *IEEE Trans Commun* 39:669–682
10. Porat B, Friedlander B (1991) Blind equalization of digital communication channels using high-order moments. *IEEE Trans Signal Process* 39:522–526
11. Cadzow JA (1996) Blind deconvolution via cumulant extrema. *IEEE Signal Process* 13(3):24–42
12. Cuiping L, Hongwei X (2001) Identification of Non-Gaussian non-minimum phase ARMA Model based on high-order cumulant method. *J Shanghai Univ* 7(5):438–441
13. Xizheng K, Xiaoli X (eds) (2004) Introduction to wireless laser communication. Beijing University of Posts and Telecommunications Press, Beijing, pp 239–244
14. Xiaoqin Z (2003) Research on blind equalization algorithm based on high order spectrum theory, vol 4. Taiyuan University of Technology, Taiyuan, pp 17–22
15. Giannakis GB (1987) Cumulants: a powerful tool in signal processing. *Proc IEEE* 75:1333–1334
16. Zhang XD, Zhang YS (1994) FIR system identification using higher cumulants alone. *IEEE Trans Signal Process* 42, 2854–2858
17. Van RM, Ness JW (1965) Estimation of the bispectrum. *Ann Math Statist* 36:1120–1136
18. Xianda Z (1994) Modern signal processing. Tsinghua University Press, Beijing, pp 263–274
19. Zheng FC et al (1993) Cumulant based deconvolution and identification: several new families of linearequations. *Signal Process* 30(2):199–219
20. Zheng FC, McLaughlin S, Mulgrew B (1993) Blind equalization of non minimum phase channels: high-order cumulant based algorithm. *IEEE Trans Signal Process* 41(2):681–691
21. Chenglin Z (1997) Research on system parameter estimation and blind equalization algorithm based on higher order statistics. Beijing University of Posts and Telecommunications, Beijing
22. Proakis JG (2005) Modern communication system (MATLAB Version), 2nd edn, Trans. Liu Shutang. Publishing House of Electronics Industry, Beijing
23. Liyi Z, Jingyi L (2007) Research on third and second order normalized cumulant blind equalization algorithm for real number systems. *J Circuits Syst* 2, 12(1):78–81
24. Xianda Z, Zheng B (2000) Communications signal processing, vol 12. National Defense Industry Press, pp 277–281
25. Bussgang JJ (1952) Cross correlation functions of amplitude-distorted Gaussian signals. Technical Report 216. MIT Research Laboratory of Electronics, Cambridge
26. Barratt J, Lampard D (1955) An expansion for some second-order probability distributions and its application to noise problems. *IRE Trans Inform Theory* 1:10–15
27. Gang L (2006) Research on blind equalization technology in communication system. Jilin University, p 8
28. Feng W (2003) Underwater acoustic channel blind equalization theory and algorithm based on higher order statistics. Xi'an: Northwestern Polytechnical Univ 5(18–20):103–104
29. Godard D (1980) Self-recovering equalization and carrier-tracking in two-dimensional data communication systems. *IEEE Trans Commun COM-28*:1867–1875.
30. Benveniste A, Goursat M (1984) Blind equalizers. *IEEE Trans Commun* 32(8):871–883
31. Deshpande N (1997) Fast recovery equalization techniques for DTV signals. *Trans Broadcast* 43(4):370–377
32. Yiping Z (2004) Design of blind equalizer based on FPGA [D], Xi'an: Northwestern Polytechnical University, pp 06–28
33. Feng CC, Hsi CH, Chi CY (1999) Performance of Shalvi and Weinstein's blind deconvolution criteria for channels with/without zeros on the unit circle. In: Proceedings of the 2nd IEEE SP workshop on SPAWC, Annapolis. Maryland, 1999, May 9–12, pp 82–85
34. Ying G, Sheng-li X (2003) A class of cumulant domain adaptive filtering algorithms. *J Guangzhou* 2(2):126–130

35. Ying G (2002) Adaptive filtering theory based on cumulant and its application, vol 4. South China University of Technology, Guangzhou, pp 24–26
36. Wangfeng S, Huiling G, Zhijun L (2005) Fault feature extraction based on high-order statistic adaptive filtering. *Signal Process* 21(5):544–547
37. Lixin C (2005) Experimental measurement of atmospheric laser communication system. Xi 'an University of Technology, Xi 'an, pp 42–47
38. Xuewen Z (2006) Study on time-domain equalization in atmospheric laser communication system. Xi 'an University of Technology, Xi 'an, pp 2–5
39. Ni W (2008) Atmospheric channel equalization based on higher order statistics. Xi 'an University of Technology, Xi 'an